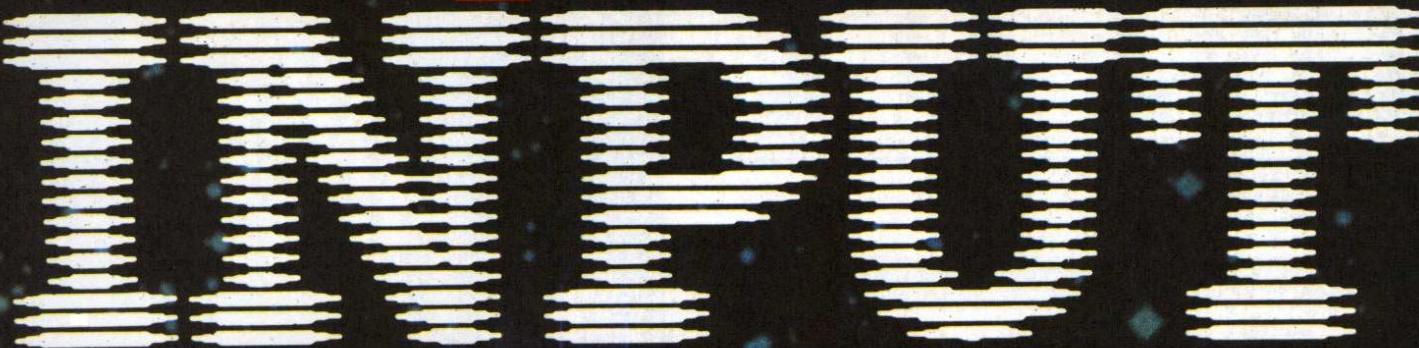


# CURSO PRÁTICO 13 DE PROGRAMAÇÃO DE COMPUTADORES



PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA



Cz\$ 20,00



# INPUT

Vol. 1

N.º 13

## NESTE NÚMERO

### PROGRAMAÇÃO BASIC

#### CADEIAS DE CARACTERES

Úteis para quem quer trabalhar com algo mais do que simples números, as cadeias de caracteres (ou cordões) são empregadas em quase todos os tipos de programas. Utilize os cordões para processamento de palavras ..... 241

### CÓDIGO DE MÁQUINA

#### ASSEMBLER PARA O SPECTRUM

Tradução de Assembly para código de máquina. Como calcular saltos e desvios. O uso de rótulos. Espaços reservados para dados e variáveis. Como colocar os códigos na memória usando comandos POKE ..... 248

### APLICAÇÕES

#### UM PROFESSOR DE DATILOGRAFIA

Aprenda datilografia com o computador e torne-se mais eficiente no processamento de textos e na digitação de programas. Saiba como utilizar as teclas de apoio e melhore a velocidade e a precisão de sua datilografia. Um programa que resolve todas essas questões ..... 253



Editor

VICTOR CIVITA

### REDAÇÃO

Diretora Editorial: Iara Rodrigues

Editor chefe: Paulo de Almeida  
Editor de texto: Cláudio A.V. Cavalcanti

Editor de Arte: Eduardo Barreto

Chefe de Arte: Carlos Luiz Batista

Assistentes de Arte: Ailton Oliveira Lopes, Dilvacy M. Santos, José Maria de Oliveira, Grace A. Arruda, Monica Lenardon Corradi

Secretária de Redação/Coordenadora: Stefânia Crema  
Secretários de Redação: Beatriz Hagström, José Benedito de Oliveira Damião, Maria de Lourdes Carvalho, Marisa Soares de Andrade, Mauro de Queiroz

Secretário Gráfico: Antonio José Filho

### COLABORADORES

Consultor Editorial Responsável: Dr. Renato M.E. Sabbatini (Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas)

Execução Editorial: DATAQUEST Assessoria em Informática Ltda, Campinas, SP.

Tradução: Maria Fernanda Sabbatini

Adaptação, programação e redação: Abílio Pedro Neto, Aluísio J. Dornellas de Barros, Marcelo R. Pires Therezo, Raul Neder Porrelli,

Coordenação geral: Rejane Felizatti Sabbatini

Assistente de Arte: Dagmar Bastos Sampaio

### COMERCIAL

Diretor Comercial: Roberto Martins Silveira

Gerente Comercial: Flávio Ferrucio Maculan

Gerente de Circulação: Denise Maria Mozol

### PRODUÇÃO

Gerente de Produção: João Stungis

Coordenador de Impressão: Atílio Roberto Bonon

Preparador de Texto/Coordenador: Eiel Silveira Cunha

Preparadores de Texto: Ana Maria Dilguerian, Antonio

Francelino de Oliveira, Karina Ap. V. Grechi, Levon Yacubian, Maria Teresa Galluzzi, Paulo Felipe Mendrone

Revisor/Coordenador: José Maria de Assis

Revisores: Conceição Aparecida Gabriel, Isabel Leite de Camargo, Ligia Aparecida Ricetto, Maria do Carmo Leme Monteiro, Maria Luiza Simões, Maria Teresa Martins Lopes.

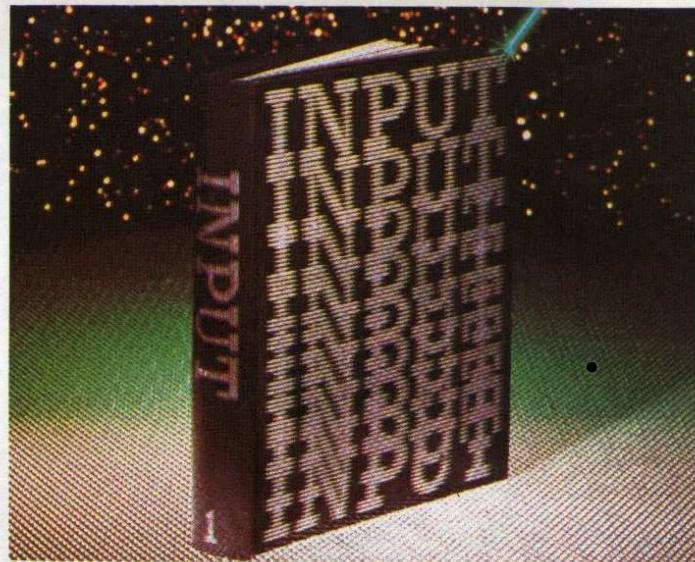
© Marshall Cavendish Limited, 1984/85.

© Editora Nova Cultural Ltda., São Paulo, Brasil, 1986.

Edição organizada pela Editora Nova Cultural Ltda.

(Artigo 15 da Lei 5 988, de 14/12/1973).

Esta obra foi composta na AM Produções Gráficas Ltda. e impressa na Dívisão Gráfica da Editora Abril S.A.



### PLANO DA OBRA

"INPUT" é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

### COMPLETE SUA COLEÇÃO

Exemplares atrasados, até seis meses após o encerramento da coleção, poderão ser comprados, a preços atualizados, da seguinte forma: 1. **Pessoalmente** — por meio de seu jornaleiro ou dirigindo-se ao distribuidor local, cujo endereço poderá ser facilmente conseguido junto a qualquer jornaleiro de sua cidade. Em São Paulo os endereços são: Rua Brigadeiro Tobias, 773 (Centro); Av. Industrial, 117 (Santo André); e, no Rio de Janeiro: Rua da Passagem, 93 (Botafogo). 2. **Por carta** — Poderão ser solicitados exemplares atrasados também por carta, que deve ser enviada para DINAP — Distribuidor Nacional de Publicações — Números Atrasados — Estrada Velha de Osasco, 132 (Jardim Tereza) — CEP 06000 — Osasco — São Paulo. 3. **Por telex** — Utilize o n.º (011) 33670 ABSA. Em Portugal, os pedidos devem ser feitos à Distribuidora Jardim de Publicações Ltd. — Qta. Pau Varais, Azinhaga de Fetais — 2685, Camarate — Lisboa; Tel. 257-2542 — Apartado 57 — Telex 43 069 JARLIS P.

Não envie pagamento antecipado. O atendimento será feito pelo reembolso postal e o pagamento, incluindo as despesas postais, deverá ser efetuado ao se retirar a encomenda na Agência do Correio. **Atenção:** Após seis meses do encerramento da coleção, os pedidos serão atendidos, dependendo da disponibilidade de estoque. **Obs.:** Quando pedir livros, mencione sempre o título e/ou o autor da obra, além do número da edição.

### COLABORE CONOSCO

Encaminhe seus comentários, críticas, sugestões ou reclamações ao Serviço de Atendimento ao Leitor — Caixa Postal 9442, São Paulo — SP.

# CADEIAS DE CARACTERES

Empregadas em quase todos os tipos de programas, as cadeias de caracteres (ou cordões) são muito úteis para quem quer trabalhar com algo mais do que simples números.

Um cordão (ou *string*, em inglês) é, como já vimos, um conjunto de caracteres. Estes, por sua vez, podem ser letras, números, sinais de pontuação ou quaisquer outros símbolos disponíveis no teclado.

Normalmente, um cordão contém uma ou mais informações úteis, ou mesmo partes de informação. Por exemplo, o cordão "PEDRO SILVA 241067 S" engloba diversos dados sobre uma única pessoa: nome, sobrenome, data de nascimento e estado civil. Esses dados são considerados como uma mesma comunicação. A data de nascimento pode ser dividida em dia, mês e ano; assim, na verdade, existem no total seis partes distintas de informação.

Em casos como o do exemplo acima, faz-se necessário fracionar o cordão para extrair as diferentes partes da comunicação. Em outros, ao contrário, pode ser necessário unir dois ou mais cordões. Há casos, ainda, em que se torna imprescindível medir a extensão de um cordão (número total de caracteres) e calcular o valor de algumas de suas partes numéricas. Tudo isso é possível

quando se utilizam certas funções e declarações disponíveis no BASIC da maioria dos computadores.

Conhecida como concatenação, a mais simples dessas operações consiste em unir cordões. Para efetuá-la, emprega-se o símbolo "+". Assim, se A\$ for igual a "TUDO" e B\$ igual a "BEM", então A\$ + B\$ será "TUDO BEM". A concatenação junta os cordões sem acrescentar nada a eles. Assim, "439" + "241" é igual a "439241" e não a 680.

- COMPARA E CLASSIFIQUE CORDÕES ALFANUMÉRICOS
- COMO FRACIONAR CORDÕES
- UTILIZE CORDÕES PARA PROCESSAMENTO DE PALAVRAS

```
UTA. QUE FRUTA SOU EU?"
90 INPUT AS
100 IF AS=BS THEN GOTO 160
110 LET G=G+1
120 PRINT "ERRADO"
130 FOR J=1 TO 200
140 NEXT J
150 GOTO 90
160 IF G=1 THEN PRINT "VOCE A CERTOU NA PRIMEIRA TENTATIVA":
STOP
170 PRINT "VOCE ACERTOU APOS "
;G;" TENTATIVAS"
180 STOP
```



```
10 G=1:ON RND(6) GOTO 20,30,40,
50,60,70
20 BS="MACA":GOTO 80
30 BS="LARANJA":GOTO 80
40 BS="BANANA":GOTO 80
50 BS="LIMAO":GOTO 80
60 BS="FRAMBOESA":GOTO 80
70 BS="ABACAXI"
80 CLS:PRINT"EU SOU UMA FRUTA.
QUE FRUTA SOUEU?"
90 INPUT AS
100 IF AS=BS THEN GOTO 160
110 G=G+1
120 PRINT"ERRADO!"
130 FOR J=1 TO 1000
140 NEXT J
150 GOTO 90
160 IF G=1 THEN PRINT"VOCE ACER
TOU EM UMA TENTATIVA" ELSE PRIN
T"VOCE ACERTOU APOS";G;"TENTATI
VAS"
```

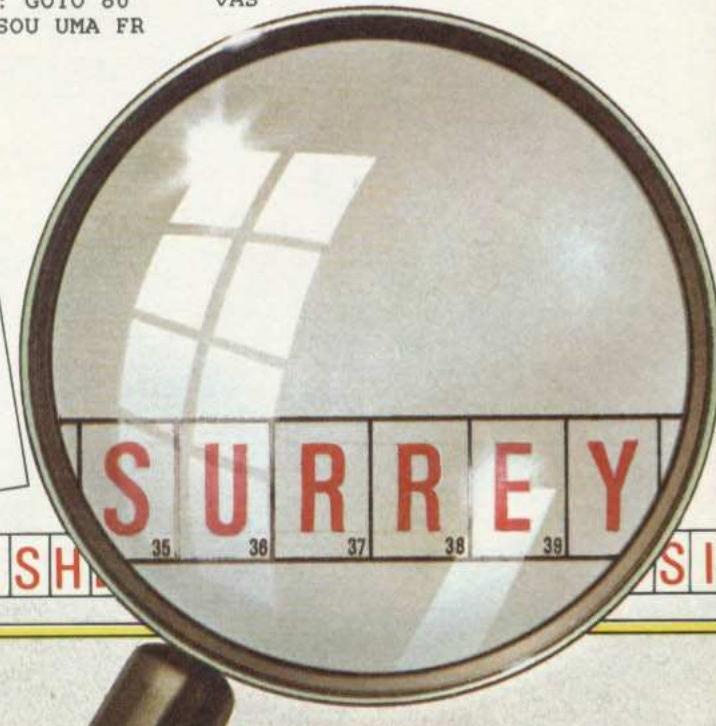
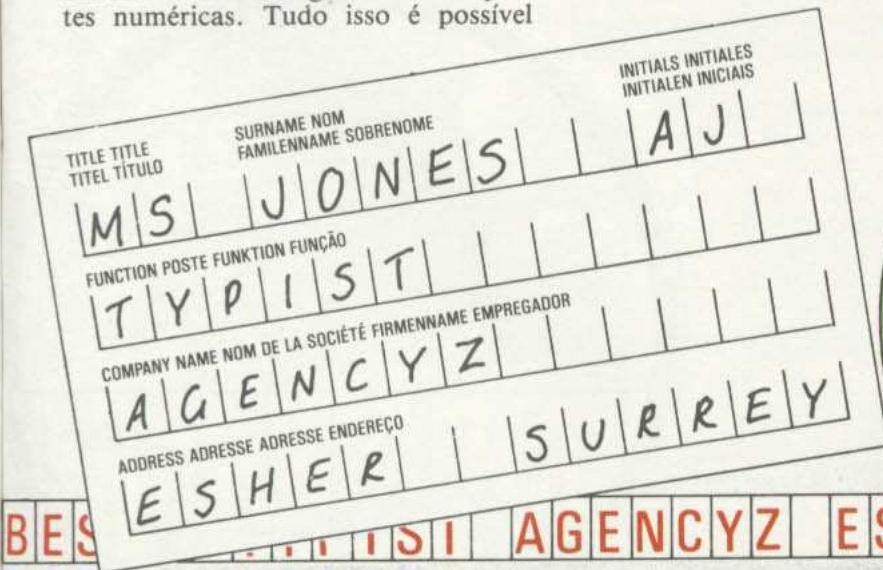
## COMO COMPARAR CORDÕES

Assim como é capaz de concatenar cordões, o computador pode compará-los para verificar se são iguais, como nesse simples jogo de adivinhação.

**SS**

Este programa funcionará no ZX-81 se você separar todas as linhas de declarações múltiplas.

```
10 LET G=1: GOTO INT (RND*6)*
10+10
20 LET BS="MACA": GOTO 80
30 LET BS="LARANJA": GOTO 80
40 LET BS="BANANA": GOTO 80
50 LET BS="LIMAO": GOTO 80
60 LET BS="FRAMBOESA": GOTO
80
70 LET BS="ABACAXI": GOTO 80
80 CLS : PRINT "EU SOU UMA FR
```





```

5 R=RND(-TIME)
10 G=1:ON INT(RND(1)*6)+1 GOTO
20,30,40,50,60,70
20 BS="MAÇÃ":GOTO 80
30 BS="LARANJA":GOTO 80
40 BS="BANANA":GOTO 80
50 BS="LIMÃO":GOTO 80
60 BS="MARACUJÁ":GOTO 80
70 BS="ABACAXI":GOTO 80
80 CLS:PRINT"Adivinhe: que frut
a sou eu?"
90 INPUT AS
100 IF AS=BS THEN 160
110 G=G+1
120 PRINT"Você errou!"
130 FOR J= 1 TO 600
140 NEXT
150 GOTO 90
160 IF G=1 THEN PRINT"Você acer
tou na primeira!" ELSE PRINT"Vo
cê tentou";G;"vezes até acertar
..."

```



Copie o programa anterior sem a linha 5 e troque **CLS** por **HOME**. Modifique a linha 160 e adicione a 170 como se segue:

```

160 IF G=1 THEN PRINT"Você acer
tou na primeira!":END
170 PRINT"Você tentou ";G;" vez
es..."

```

Aqui, a linha 10 coloca o contador de palpites em 1 e "lança" em seguida um dado eletrônico, que serve para selecionar uma fruta, ao acaso. As opções são armazenadas nas linhas 20 a 70. Qualquer que seja a linha que o computador passe, ele armazena a fruta como um cordão em **BS** e vai para a linha 80. Nesse ponto você tem que adivinhar qual fruta foi sorteada e digitar seu palpite, que será armazenado em **AS** e comparado com **BS**. Se estes forem iguais, o computador imprimirá: "Você acertou na primeira!", ou "Você tentou (tantas) vezes até acertar...". Se **AS** não for igual a **BS** o computador imprimirá: "Você errou"; nesse caso, uma nova tentativa deve ser feita.

Mesmo que você adivinhe o nome da fruta, a condição **AS = BS** não será satisfeita se você escrevê-lo de modo incorreto. Para que o computador considere dois cordões iguais, eles devem ser graficamente idênticos — isto é, letras, espaços, sinais de pontuação e números devem ser iguais.

Uma das funções dessa técnica é verificar entradas:

```
IF AS= "SIM" THEN PRINT "VOCÊ
TEM CERTEZA?"
```

A condição não será satisfeita se a palavra "sim" for digitada com letras minúsculas.

### ORDENAÇÃO DE CORDÕES

As cadeias de caracteres também podem ser comparadas utilizando-se os sinais de desigualdade < e > em linhas como essa:

```
IF AS<BS THEN PRINT "O PRIMEIRO
É ";AS
```

Aqui, a condição **AS < BS** pergunta se o cordão em **AS** vem antes do cordão em **BS**, quando eles são colocados em ordem alfabética. Mas tenha cuidado: o computador coloca cordões em ordem alfabética, examinando o código ASCII de cada uma das letras ou caracteres que compõem o cordão; por exemplo, a letra A no código ASCII equivale a 65 e Z, a 90. O problema é que as letras minúsculas também possuem códigos ASCII: o a é 97 e o z, 122. Assim, todos os cordões que começarem com letras maiúsculas serão colocados em primeiro lugar.

Para complicar ainda mais, sinais de pontuação, espaços e outros signos também possuem códigos ASCII; dessa forma é difícil prever em que ordem ficará um cordão constituído por diversos signos.

Com o devido cuidado, os operadores de maior (>) e menor (<), ainda podem ser utilizados para organizar cordões em ordem alfabética. Uma rotina de ordenação alfabética, que aplica a chamada técnica de ordenação por bolhas, será explicada mais adiante numa lição sobre estruturação de programas.

### COMO FRACIONAR CORDÕES

É possível também separar um caractere ou uma sequência de caracteres de um cordão. Nos computadores TRS, MSX e Apple II isso é feito utilizando-se funções **LEFT\$**, **RIGHT\$** e **MID\$**. Já o Spectrum aplica uma técnica diferente.

A função **LEFT\$ (A\$, número)** toma caracteres a partir do primeiro à esquer-

da no cordão **AS** e fornece tantos caracteres quantos forem especificados no número entre parênteses. Se **AS** for "SR MARIO SILVA" e você especificar dois caracteres — **LEFT\$ (A\$,2)** —, o resultado será "SR".

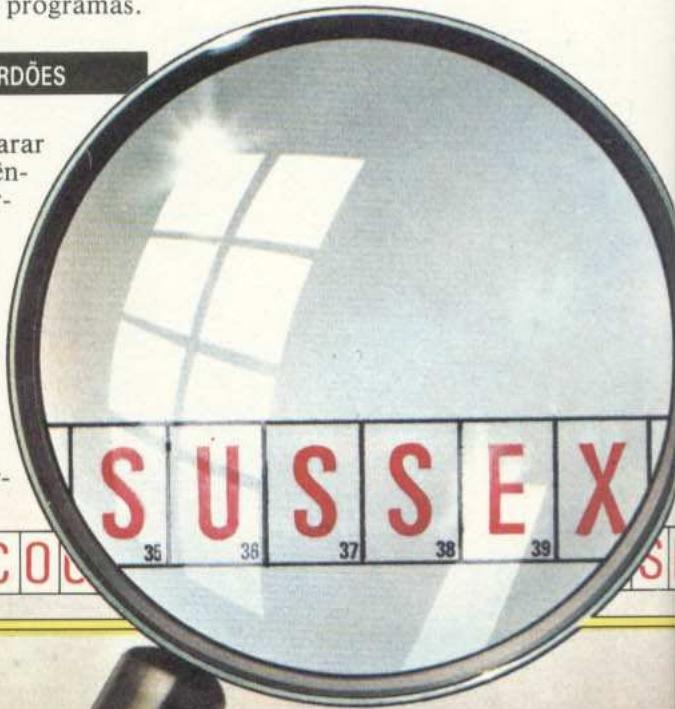
De igual modo, o comando **RIGHT\$** conta a partir da outra ponta do cordão — ou seja, de seu final à direita. Assim, **RIGHT\$ (A\$,5)** produzirá "SILVA".

Com o **MID\$** você pode especificar dois parâmetros: a posição inicial no cordão, e o número de caracteres a serem extraídos. Por exemplo, **MID\$ (A\$,4,5)** começará no quarto caractere **M** e pegará cinco caracteres, produzindo "MARIO". Se for especificado somente um número, tal como em **MID\$(A\$,4)**, a função retornará todos os caracteres daquela posição, até o final do cordão.

O método empregado pelos micros da linha Sinclair, tais como o ZX-81 e o ZX Spectrum, é mais simples e direto. Nesses computadores, existe apenas uma função para manipulação de cordões: o comando **TO**. Por exemplo, **AS (número TO número)**. **AS** identifica o cordão a ser fracionado e os dois parâmetros correspondentes ao início e ao fim do sub-cordão a ser extraído.

Usando o mesmo cordão do exemplo anterior, **AS = "SR MARIO SILVA"**, **AS(1 TO 2)** dá a você "SR", **AS (4 TO 8)** fornece "MARIO" e **AS(10 TO 14)** produz "SILVA". Não é necessário especificar os números na função acima. Se for omitido o primeiro, o sub-cordão a ser extraído se iniciará no primeiro caractere. Se o segundo for omitido, a função extrairá até o final do cordão.

Eis aqui um programa que utiliza todas as funções explicadas acima. É um jogo de palavras para duas pessoas.



ES DETYPIST AGENCYZ COU

Uma delas entra uma expressão cujas letras serão embaralhadas e impressas pelo computador como um anagrama para o segundo jogador resolver.

**T**

```

10 CLS
20 PRINT @75, "ANAGRAMA"
30 PRINT @161, "DIGITE A PALAVRA
A SER EMBARA LHADA"
40 AS=INKEY$: IF AS="" THEN 40
43 IF AS=CHR$(13) THEN 55
46 IF AS<" " THEN 40
49 WS=WS+AS: GOTO 40
55 WORDS=WS
70 CLS
80 FOR N=LEN(WS) TO 1 STEP -1
90 M=RND(N)
100 AS=AS+MIDS(WS,M,1)
110 WS=LEFTS(WS,M-1)+MIDS(WS,M+
1)
120 NEXT N
130 PRINT @65, "O ANAGRAMA E "AS
140 PRINT @129, "QUE PALAVRA VOC
E ACHA QUE E?"
160 INPUT GUESS$
170 G=G+1
180 IF GUESS$<>WORDS THEN PRINT
" ERRADO, TENTE OUTRA VEZ": GOTO
160
190 PRINT:PRINT" MUITO BEM !"
200 IF G=1 THEN PRINT" VOCE USO
U 1 TENTATIVA" ELSE PRINT"VOCE
USOU";G;"TENTATIVAS"
210 PRINT @480, "QUER JOGAR DE N
OVO (S/N)?"
220 AS=INKEY$: IF AS<>"S" AND AS
<>"N" THEN 220
230 IF AS="S" THEN RUN
240 END

```



```

10 HOME
20 HTAB 5: VTAB 2: PRINT "PROG
RAMA DE ANAGRAMAS"
30 VTAB 5: PRINT "DIGITE UMA P
ALAVRA PARA SER MISTURADA:"
50 INPUT WS
55 WOS = WS
70 HOME
80 FOR N = LEN (WS) TO 1 STEP
- 1
90 M = INT ( RND ( 1 ) * ( N - 1 )
) + 1
100 AS = AS + MIDS ( WS, M, 1 )
105 IF M = 1 THEN WS = MIDS (
WS, M + 1): GOTO 120
110 WS = LEFTS ( WS, M - 1 ) + M
IDS ( WS, M + 1 )
120 NEXT
130 HTAB 4: VTAB 12: PRINT "O
ANAGRAMA E: ";AS
140 PRINT : HTAB 4: PRINT "QUE
PALAVRA E ESTA ";
160 INPUT GUESS$
170 G = G + 1
180 IF GUESS$ < > WOS THEN P
RINT : PRINT "ERRADO! TENTE NOV
AMENTE.": GOTO 160
190 PRINT : PRINT "MUITO BEM !"

```

```

200 IF G = 1 THEN PRINT "VOCE
ACERTOU NA PRIMEIRA!": GOTO 21
0
205 PRINT "VOCE TENTOU ";G;" V
EZES."
210 PRINT : PRINT : PRINT "VOC
E JOGA OUTRA VEZ? "
220 GET AS: IF AS < > "S" AND
AS < > "N" THEN 220
230 IF AS = "S" THEN RUN
240 END

```



```

10 CLS: R=RND(-TIME)
20 LOCATE 5,2:PRINT"
PROGRAMA DE ANAGRAMAS"
30 LOCATE 1,5:PRINT"
Digite uma palavra para
ser misturada:"
50 INPUT WS
55 WOS=WS
70 CLS
80 FOR N=LEN(WS)TO1STEP-1

```

```

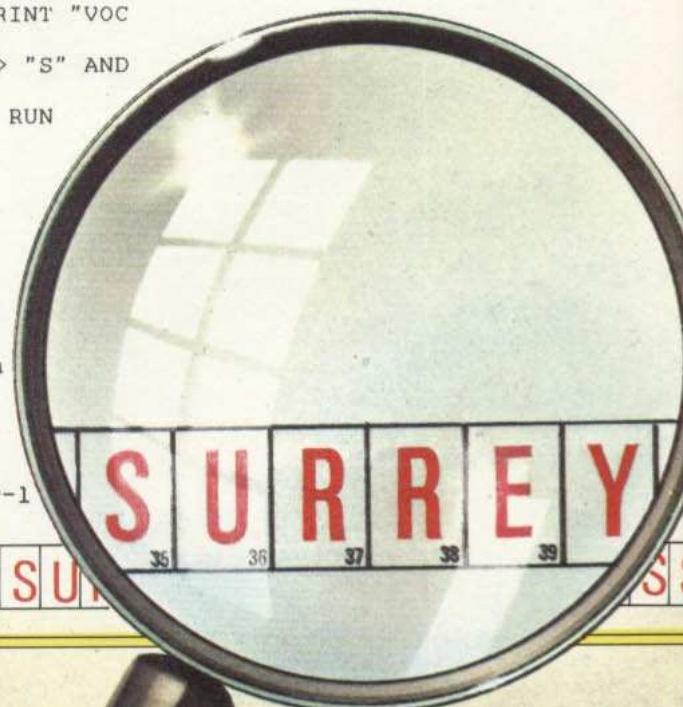
90 M=INT(RND(1)*(N-1))+1
100 AS=A$+MIDS(WS,M,1)
110 WS=LEFTS(WS,M-1)+MIDS(WS,M+
1)
120 NEXT
130 LOCATE 4,12:PRINT"O anagram
a é: ";AS
140 LOCATE 4,14:PRINT"Que palav
ra é esta? "
160 INPUT GUESS$
170 G=G+1
180 IF GUESS$<>WOS THEN PRINT "
Errado! Tente novamente.":GOTO
160
190 PRINT:PRINT" Muito bem !"
200 IF G=1 THEN PRINT"Vocé acert
ou na primeira!" ELSE PRINT"Voc
é tentou";G;"vezes."
210 PRINT:PRINT:PRINT"Você joga
outra vez? (S/N)"
220 AS=INKEY$: IF AS<>"S" AND AS
<>"N" THEN 220
230 IF AS= "S" THEN RUN
240 END

```

Quando rodar esse jogo, você verá que a palavra digitada em primeiro lugar não aparecerá na tela. Esse ocultamento tem por objetivo evitar que o seu adversário veja qual é a palavra. Cada computador utiliza um método diferente para fazer isso. Os micros da linha Spectrum imprimem a palavra na mesma cor do plano de fundo, de modo a torná-la invisível.

O TRS-Color entra a palavra utilizando **INKEY\$**, que apanha um caractere por vez no teclado, sem imprimi-los na tela. O MSX e o Apple II simplesmente apagam a tela depois da digitação da palavra.

A rotina para embaralhar as letras está nas linhas 80 a 120. O que acontece é que os caracteres são apanhados aleatoriamente na palavra e acrescentados



URST ATTYPYST AGENCYZ SU

SURVEY  
35 36 37 38 39

(um de cada vez) a A\$ que, gradualmente, monta o anagrama.

A linha 90 escolhe um número aleatório M, entre 1 e a extensão da palavra. A linha 100 pega a M-ésima letra e a acrescenta a A\$. A linha 110 remove esse caractere da expressão original, tomando a parte esquerda da palavra até a posição M (exclusive) e acrescentando o que vier após M. Agora, a expressão terá um caractere a menos, mas da próxima vez, ao fechar o laço, a variável N também terá seu valor reduzido de um; assim, o número aleatório M será novamente limitado pela extensão da palavra.

Quando todos os caracteres tiverem sido extraídos, o anagrama será impresso na tela e o seu adversário terá que adivinhar qual a palavra original. Quando ele acertar, o computador informará sobre o número de tentativas feitas durante o jogo e oferecerá uma nova partida.

É bastante fácil alterar esse programa, de modo a fazer com que as palavras sejam lidas a partir de uma lista de dados em linhas DATA, em vez de entrá-las separadamente a todo momento. Você pode igualmente elaborar um sistema de contagem de pontos, atribuindo, por exemplo, dez pontos para uma adivinhação correta feita na primeira tentativa, nove para uma adivinhação correta após duas tentativas e assim por diante.

Uma outra utilização desse método de fracionamento é a manipulação de datas. Estas formam um cordão, mesmo quando são digitadas em forma numérica — como, por exemplo, 27/03/51, que significa 27 de março de 1951. É bem verdade que você não pode manipular uma data expressa desse modo pelas leis normais da matemática. Mas pode trabalhar com diferentes partes de uma data utilizando aritmética.

Dessa maneira, é possível, por exemplo, calcular quantos anos uma pessoa tem em um determinado dia, desde que se conheça a data de seu nascimento. Da mesma forma, pode-se calcular quantos dias se passaram entre duas datas determinadas.

Os comandos LEFT\$, RIGHTS e MIDS — ou os métodos equivalentes usados pelos micros da linha Sinclair — podem ser usados facilmente para separar o dia, o mês e o ano de uma data. Caso seja empregada a função VAL (explicada mais adiante), os cordões contendo números poderão ser convertidos

para números, que podem ser manipulados por meio de operações aritméticas tais como divisão, soma, subtração, etc.

### O TAMANHO DE UM CORDÃO

As vezes é útil conhecer o comprimento de um cordão. Se, por exemplo, você tiver um espaço limitado de memória, reservado para a informação digitada no teclado, ou se dispuser de uma área restrita na tela para exibi-la, pode ser conveniente verificar a extensão da entrada antes de continuar com o programa.

O comando LEN(A\$) fornece o número de caracteres de um cordão A\$. É uma função numérica e pode ser manipulada de acordo com as leis da álgebra. Por exemplo, se A\$ = "SR JOÃO SILVA", LEN(A\$) = 13. Mas, se o programa exigir que o usuário entre um nome, e só houver espaço suficiente na tela para exibir onze letras, o trecho seguinte poderá ser utilizado para informar ao usuário que ele deve limitar a extensão do nome a ser entrado, eliminando letras ou mesmo palavras:



```
10 PRINT "DIGITE O NOME DO ASSUNTO"
20 INPUT AS
30 IF LEN(A$)>11 THEN PRINT
  "APENAS 11 CARACTERES DISPONIVEIS": GOTO 10
```

O usuário deverá então reduzir sua entrada e teclar "JOÃO SILVA".

Em outros casos, pode ser mais fácil truncar automaticamente uma entrada. Isso é feito com uma linha tal como a que se segue:



```
IF LEN(A$)>15 THEN LET AS=LEFT$(AS,15)
```



```
IF LEN(A$)>15 THEN
  LET AS=A$(TO 15)
```

### NÚMEROS

Uma das aplicações mais importantes das funções de conversão de cordões para números

consiste em fazer programas "à prova de idiotas", ou seja, que impeçam a digitação equivocada dos dados. Por exemplo, na seção abaixo:

```
100 PRINT "DIGITE UM NUMERO"
110 INPUT A
```

...o computador espera que o usuário digite um número. Se este digitar um caractere não numérico, a maioria dos computadores imprimirá uma mensagem de erro. Esta, porém, avisará apenas que alguma coisa está errada, sem dizer onde está o erro.

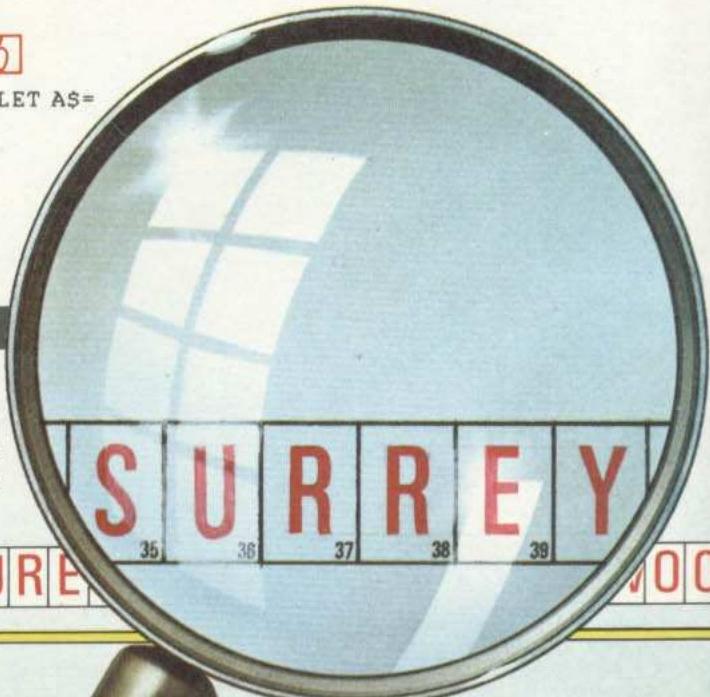
Porém, se for utilizado:

```
110 INPUT AS
```

...o computador aceitará qualquer coisa digitada pelo usuário sem emitir mensagens de erro. O passo seguinte será converter o cordão em um número. Para isso, utiliza-se a função VAL(). Na maioria dos computadores, se houver alguma letra misturada com números no cordão de entrada, VAL retornará um valor igual a 0.

Infelizmente, esse método não serve para os micros da linha Sinclair, nos quais o emprego do comando VAL com um cordão de caracteres que contenha letras produz uma mensagem de erro. Nesses micros, a função VAL só deve ser usada se o cordão for constituído apenas de números. Assim, VAL("1984") fornecerá 1984; mas VAL("26/10/84") fornecerá 0.03095..., porque os micros da linha Sinclair utilizam o VAL para avaliar a expressão 26 : 10 : 84. Por esse motivo, os possuidores do Sinclair não precisam ler o resto desta seção, dirigindo-se diretamente para o próximo "capítulo" do artigo (*De números a cordões*).

Nos computadores TRS-80, TRS-



AMTYPYST AGENCYZ SURE

S U R V E Y

35 36 37 38 39

700

Color, Apple e MSX, a função **VAL** extrai a parte numérica do cordão. Assim, se **A\$** contiver apenas dígitos numéricos, **VAL(= A\$)** retornará o valor numérico correspondente, que poderá ser posteriormente utilizado no programa. Se, ao contrário, **A\$** não contiver números, **VAL(A\$)** retornará o valor 0.

Você poderá então escrever uma pequena sub-rotina que explique em detalhes ao usuário que ele cometeu um erro, oferecendo-lhe ao mesmo tempo outra oportunidade de entrar os dados sem interromper o programa ou desarrumar a tela do computador.

Um ponto importante a ser observado em relação ao comando **VAL** é que ele só extrai números no início do cordão. Dessa maneira, **VAL("25 Julho")** é igual a 25, mas **VAL("JULHO 25")** é equivalente a 0.

A função **VAL** é útil também para ordenar números extraídos de um cordão. Se, por qualquer razão, você tiver os nomes e as notas de uma classe de alunos em um cordão como **A\$ = "32 CARLOS"**, **B\$ = "45 MARCOS"** e **C\$ = "41 JOSÉ"** e assim por diante, e quiser uma média dos resultados, o primeiro passo a ser dado será extraír as notas individuais utilizando o **VAL**. Assim, **VAL(= A\$)** fornece 32, **VAL(= B\$)** extrai 45 e **VAL(C\$)** separa 41.

Do mesmo modo, o comando **VAL** pode ser utilizado para ignorar as unidades de medida digitadas junto com os valores. O próximo programa mostrará a você a diferença:



```
100 LET A$ = "32KG"
110 LET B$ = "110KG"
120 PRINT A$ + B$ = " ; AS + BS
130 PRINT "VAL (A$) + VAL (B$) = " ; VAL
(A$) + VAL (B$)
140 END
```

### DE NÚMEROS A CORDÕES

A função **STR\$** desempenha um papel oposto ao do comando **VAL**: enquanto este extraí números de cordões, aquela converte números em cordões. A vantagem dessa operação é que os cordões podem ser, por exemplo, fracionados e concatenados.

O comando **STR\$** possui, portanto, inúmeras aplicações.

O programa a seguir transforma um número decimal em binário. Embora os computadores trabalhem diretamente

com a aritmética de números binários, a linguagem BASIC é capaz de manipular apenas decimais ou, em alguns casos, hexadecimais e octais. Assim, quando um número binário aparecer em um programa em BASIC, ele precisa ser tratado como se fosse um cordão:



```
10 PRINT "DECIMAL PARA BINARIO"
20 PRINT "DIGITE NUMERO DECIMA
L INTEIRO"
30 INPUT D
40 LET B$ = ""
50 LET B$ = STR$(D - INT(D/2) * 2) + B$
60 LET D = INT(D/2)
70 IF D < 0 THEN GOTO 50
80 PRINT "O NUMERO BINARIO E" ; B$
```

Quando for entrado um número decimal positivo, a linha 40 igualará **B\$** a um cordão vazio ou nulo; este será então progressivamente preenchido com dígitos obtidos pelo processamento nas linhas 50 a 80.

Na verdade, a linha 50 é a que monta o número binário. Ela subtrai duas vezes o valor inteiro da metade do valor decimal, a partir do próprio número decimal. Essa é uma maneira de testar se o número é ímpar ou par. Se for ímpar, o resultado da operação será 1; se for par, o resultado será 0. Estes são, como você já sabe, os dois dígitos binários.

Esses dígitos são convertidos em um cordão mediante o emprego da função **STR\$**; em seguida, eles são montados por intermédio da concatenação de cada novo dígito com o resto do cordão.



### STRINGS E INSTR

Os computadores MSX, TRS-80 e TRS-Color contam com duas funções adicionais de manipulação de cordões alfanuméricicos:

**STRING\$** e **INSTR** (estas não existem nos micros da linha Apple, TK-2000 e Sinclair). O **STRING\$(N,A\$)** produz um cordão de **N** caracteres de-

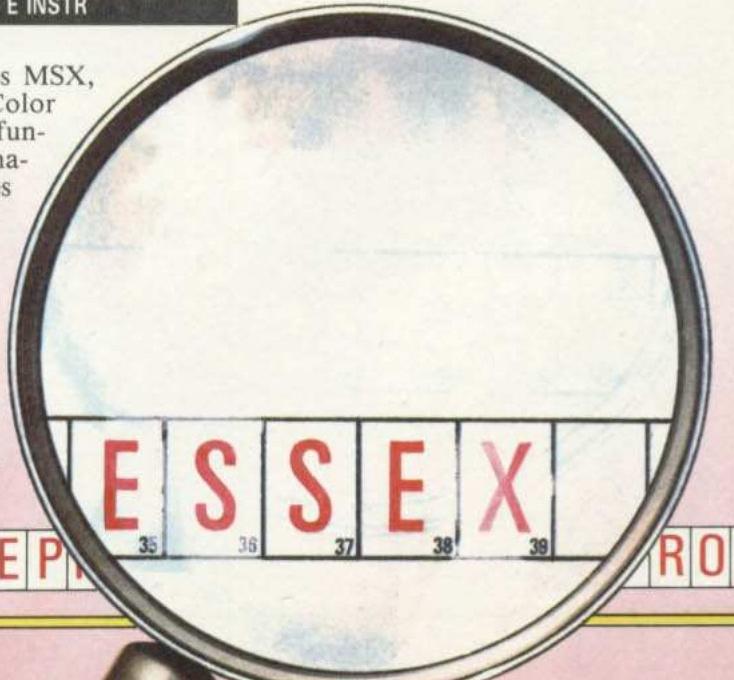
finidos por **A\$**. **N** deve ser um número ou uma variável numérica, enquanto **A\$** deve ser uma variável alfanumérica, um caractere ou um cordão de caracteres entre aspas. Por exemplo, **PRINT STRING\$(6, "")** imprimirá \*\*\*\*\*.

Nos microcomputadores das linhas TRS-80 e TRS-Color a função **STRING\$** utiliza somente o *primeiro* caractere do cordão especificado, ignorando o resto. Por exemplo, **STRING\$(3,"XBC")** imprime XXX, apenas.

Eis aqui dois programas (um para os modelos compatíveis com a linha TRS-Color e o outro para os da linha MSX) que utilizam **STRING\$** para imprimir uma borda decorativa na tela. Você pode utilizá-los para abrillantar a página-título de um jogo:



```
10 CLS0
20 A$ = CHR$(158) + STRING$(30, CHR$(156)) + CHR$(157)
30 B$ = CHR$(154) + CHR$(174) + STRING$(28, CHR$(172)) + CHR$(173) + CHR$(149)
40 C$ = CHR$(154) + CHR$(171) + STRING$(28, CHR$(163)) + CHR$(167) + CHR$(149)
50 D$ = CHR$(155) + STRING$(30, CHR$(147)) + CHR$(151)
60 F$ = CHR$(154) + CHR$(170) + STRING$(28, " ") + CHR$(165) + CHR$(149)
70 PRINT A$;
80 PRINT B$;
90 FOR K = 1 TO 11
100 PRINT F$;
110 NEXT K
120 PRINT C$;
130 PRINT D$;
140 PRINT @237, "OLA !";
150 GOTO 150
```



S LETYPYST AGENCYZ EP 35 36 37 38 39 ROSE



```

10 CLS:COLOR 6,11
20 A$=CHR$(219)+STRINGS(36,210)
+CHR$(219)
30 B$=CHR$(209)+CHR$(32)+STRING
$(34,192)+CHR$(32)+CHR$(209)
40 C$=CHR$(209)+CHR$(32)+STRING
$(34,195)+CHR$(32)+CHR$(209)
50 D$=CHR$(209)+CHR$(222)+STRIN
GS(34,32)+CHR$(221)+CHR$(209)
70 PRINTAS
80 PRINTBS
90 FORK=1TO18
100 PRINTDS
110 NEXT
120 PRINTCS
130 PRINTAS
140 LOCATE 16,10:PRINT"OLA";
150 GOTO 150

```

Os programas para o TRS-Color empregam diversos caracteres para imprimir as bordas. Os blocos são impressos com esquema de duas cores — amarelo/preto e azul/preto. Para se fazer uma borda simétrica, as cores são invertidas, de cima para baixo, e da esquerda para a direita. Cinco cordões são utilizados: **A\$** e **B\$** definem a parte superior da borda, **F\$** os lados, e **C\$** e **D\$** a margem inferior. A linha 150 impede que o sinal de prontidão OK perturbe o resultado final na tela. O programa para o MSX também utiliza caracteres gráficos para fazer as bordas.

#### COMO PESQUISAR UM CORDÃO

O comando **INSTR** examina o cordão, procurando pela primeira ocorrência de um sub-cordão mais curto. Assim, uma das aplicações mais comuns para o **INSTR** consiste em encontrar uma palavra dentro de uma sentença, ou uma letra dentro de uma palavra.

Para usá-lo, especifica-se **INSTR(A\$, B\$)**: isto significa que a função deve indicar a posição inicial do sub-cordão **B\$** no cordão **A\$**. Por exemplo, **PRINT INSTR("ALO", "L")** exibirá o número 2. Se o computador não conseguir achar o cordão, ele retornará 0:

```

10 A$="ALO"
20 B$="W"
30 PRINT INSTR(A$,B$)

```

Em alguns micros é possível ainda especificar um número que define a posição no cordão maior, a partir da qual a função **INSTR** funcionará. Assim, em **INSTR(P,A\$,B\$)**, essa posição é indi-

cada por **P**. Essa forma de utilização é muito útil quando se deseja procurar ocorrências repetidas do mesmo sub-cordão no cordão principal.

Suponhamos que se deseja encontrar todas as ocorrências da letra I na palavra **INCONSTITUCIONAL**. O primeiro I está na posição 1. Para achar o segundo, entretanto, devemos usar o parâmetro **P**, de modo a começar a busca a partir de **P = 1 + 1** (ou seja, **P** igual a 2); portanto, **INSTR(P,"INCONSTITUCIONAL","I")**. Para localizar a terceira ocorrência do I, o número **P** deve ser incrementado mais uma vez, e assim por diante, até que um resultado 0 seja retornado pela função.

Esse comando serve ainda para verificar uma entrada feita pelo teclado. Se for preciso escolher uma opção em um menu, digitando a sua letra inicial, as alternativas poderão ser:

```

10 PRINT "(I)MPRIMIR O TEXTO"
20 PRINT "(G)RAVAR O TEXTO"
30 PRINT "(C)ARREGAR NOVO
TEXTO"
40 PRINT "(E)DITAR O TEXTO"
50 PRINT "ESCOLHA UMA OPÇÃO"
60 INPUT A$

```

O modo mais fácil de se verificar a validade de uma letra digitada pelo usuário consiste em acrescentar a linha a seguir:

```

70 IF INSTR("PSLE",A$)=0 THEN
GOTO 50

```

Isso elimina possíveis mensagens de erro, que "estragariam" a tela.

#### PROCESSAMENTO DE PALAVRAS

As funções alfanuméricas do BASIC têm um grande número de aplicações, especialmente no que se refere ao processamento de palavras. Um aspecto comum a programas desse tipo é a possibilidade de substituir uma determinada palavra por outra (isso pode ser necessário, por exemplo, quando aparecer um erro de ortografia em todas as

ocorrências de uma palavra). O **INSTR** é usado para localizar essas ocorrências. Se a nova palavra tiver um comprimento diferente daquela a ser substituída, será necessário movimentar o resto do texto para criar espaço.



```

10 LINEINPUT "DIGITE TEXTO: ";T$
20 LINEINPUT "PALAVRA A SER TRO
CADA? ";WS
30 LINEINPUT "NOVA PALAVRA? ";N
WS
40 P=1
50 PO=INSTR(P,TS,WS)
60 IF PO=0 THEN GOTO 100
70 TS=LEFT$(TS,PO-1)+NWS+RIGHT$(TS,LEN(TS)-PO-LEN(WS)+1)
80 P=PO+LEN(NWS)
90 GOTO 50
100 PRINT TS
110 GOTO 20

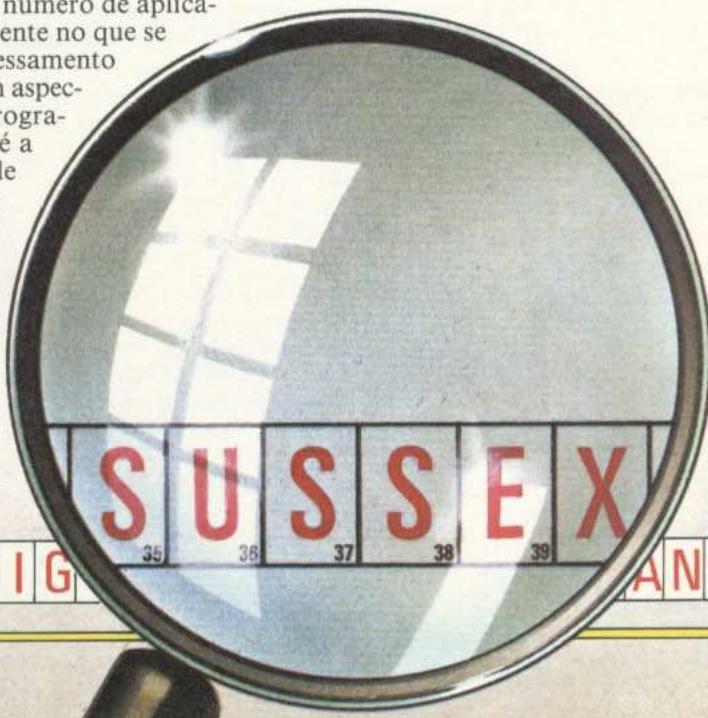
```



```

10 INPUT "DIGITE UMA FRASE ";T$
15 TS = CHR$(32) + TS + CHR$(32)
20 INPUT "PALAVRA A SER SUBSTI
TUIDA ";WS
30 INPUT "NOVA PALAVRA ";NWS
35 P = 0
40 P = P + 1
50 A$ = MID$(TS,P,LEN(WS))
60 IF A$ < > WS THEN 90
70 TS = LEFT$(TS,P - 1) + NWS
+ RIGHTS$(TS,LEN(TS) - P -
LEN(WS) + 1)
80 P = P + LEN(NWS) - 1
90 IF P < LEN(TS) THEN 40
100 PRINT TS
110 GOTO 20

```



GETYPIST AGENCYZ BRIG 35 36 37 38 39 ANN

S

```

10 INPUT "DIGITE O TEXTO ";
LINE t$: PRINT t$
20 INPUT "PALAVRA A SER CORRIDA "; LINE w$: LET w=LEN w$
30 INPUT "NOVA PALAVRA ";
LINE n$: LET n=LEN n$
35 LET p=0
40 LET p=p+1
50 IF p+w-1>LEN t$ THEN GOTO 100
60 IF t$(p TO p+w-1)<>w$ THEN
    GOTO 40
70 LET t$=t$( TO p-1)+n$+t$(p+w TO ): GOTO 40
100 PRINT t$
110 GOTO 20

```

W

```

10 LINEINPUT"Digite uma frase:
";T$
20 LINEINPUT"Palavra a ser substituída? ";WS
30 LINEINPUT"Nova palavra? ";NW
$ 
40 P=1
50 PO=INSTR(P,T$,WS)
60 IF PO=0 THEN 100
70 T$=LEFTS(T$,PO-1)+NWS+RIGHTS(T$,LEN(T$)-PO-LEN(WS)+1)
80 P=PO+LEN(NWS)
90 GOTO 50
100 PRINT T$
110 GOTO 20

```

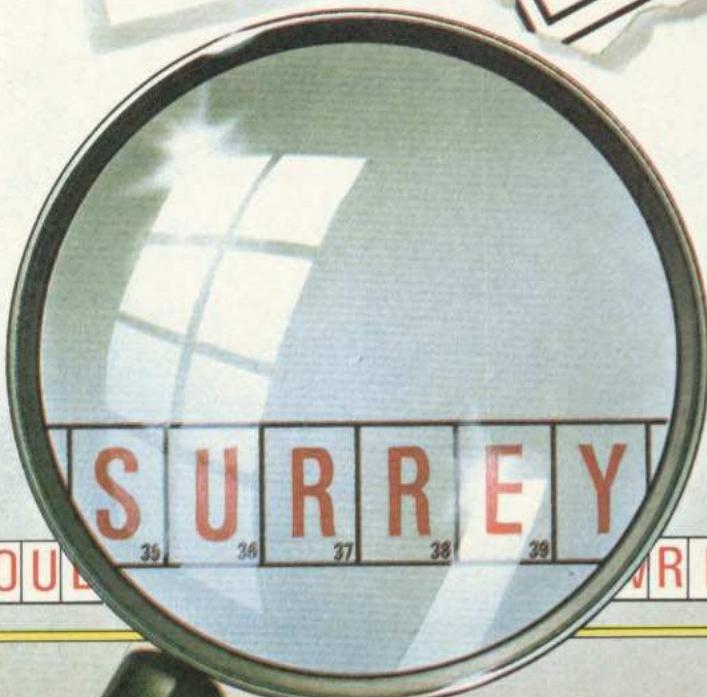
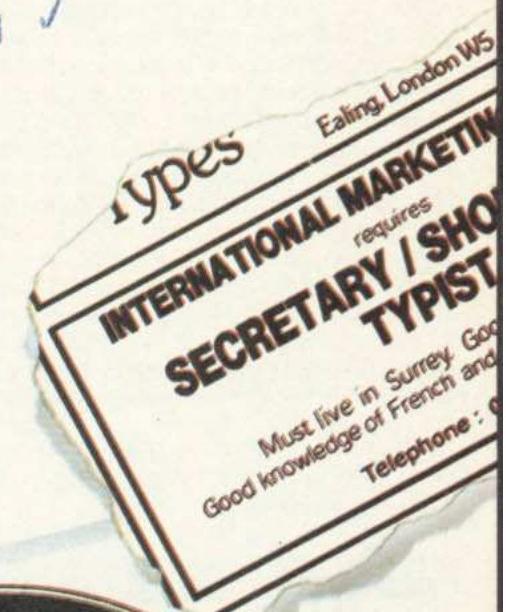
Comece entrando sentenças bem simples e tente o procedimento de substituição de algumas letras ou palavras. Quando estas forem curtas como "ou", "na", etc., será necessário digitá-las com um espaço antes e outro depois; caso contrário, toda ocorrência de "ou", "na", etc., dentro de outras palavras, (por exemplo, "ouro" e "banana") também será modificada.

O programa funciona assim: a linha 40 define o **P** para começar o processo de pesquisa no início do texto. As linhas 50 e 60 encontram a primeira ocorrência da palavra ou da letra que se deseja substituir; em seguida, a linha 70 a substitui por uma nova palavra. Essa linha investiga o texto do início até a palavra a ser modificada, acrescenta a nova expressão e concatena o restante do texto. O processo se repete até que todas as ocorrências da palavra tenham sido substituídas. A linha 100 imprime o resultado (os programas profissionais de processamento de palavras são, evidentemente, mais complexos do que nosso exemplo).

AGENCY Z

Surrey

Miss Jones ✓  
 Miss Innes  
 Miss Hurst ✓  
 Miss Smith ✓  
 Miss Woods  
 Miss Frost  
 Miss Mann ✓



# ASSEMBLER PARA O SPECTRUM

Montar um programa em linguagem de máquina calculando os códigos a mão pode ser bastante cansativo, mesmo que se saiba de cor os códigos mnemônicos e seus equivalentes hexadecimais e se esteja familiarizado com todos os modos de endereçamento. Com efeito, além de estar sujeito a erro, o trabalho de tradução e de transferência do programa para o computador é sempre tedioso. A solução está em entregar esse trabalho ao próprio computador.

Nesta lição apresentamos um programa Assembler para os micros compatíveis com o Sinclair Spectrum (por exemplo, o TK-90X) com um mínimo de 48K de memória. Versões para outros computadores serão abordadas nas próximas lições desta série. O TK-2000 não necessita de um programa desses, pois já vem com um mini-Assembler. O programa apresentado aqui não funciona no Spectrum de 16K, nem nos modelos compatíveis com o ZX-81. Ele está escrito em BASIC, o que o torna bem mais lento que os Assemblers convencionais, elaborados em código de máquina. O importante, porém, é que ele não só funciona, como é capaz de montar programas Assembly, publicados por nós ou obtidos de outras fontes. Um programa longo levará um bom tempo para ser montado.

## O ASSEMBLER

### S

```
5000 DIM k$(110,4): DIM k(110):
DIM m(110): LET h$="0123456789
ABCDEF": LET b$"": LET g$="012
3456789abcdef"
5010 DIM t$(110,24): DIM r(100)
: DIM z$(100,6): DIM z(100)
5020 DIM b(9): LET b(1)=1: FOR
i=2 TO 9: LET b(i)=b(i-1)+b(i-1)
: NEXT i
5030 DIM r$(8,4,4): FOR j=1 TO
4: FOR i=1 TO 8: READ r$(i,j):
NEXT i: NEXT j
5040 DATA "0","1","2","3","4",
"5","6","7","nz","z","nc","c","p
o","pe","p","m","0","8","16","2
4","32","40","48","56","h1","ix
","iy","bc","de","h1","sp"," "
5050 DIM s$(8,2,4): DIM t(18):
DIM u$(18,10): FOR j=1 TO 2: FO
```

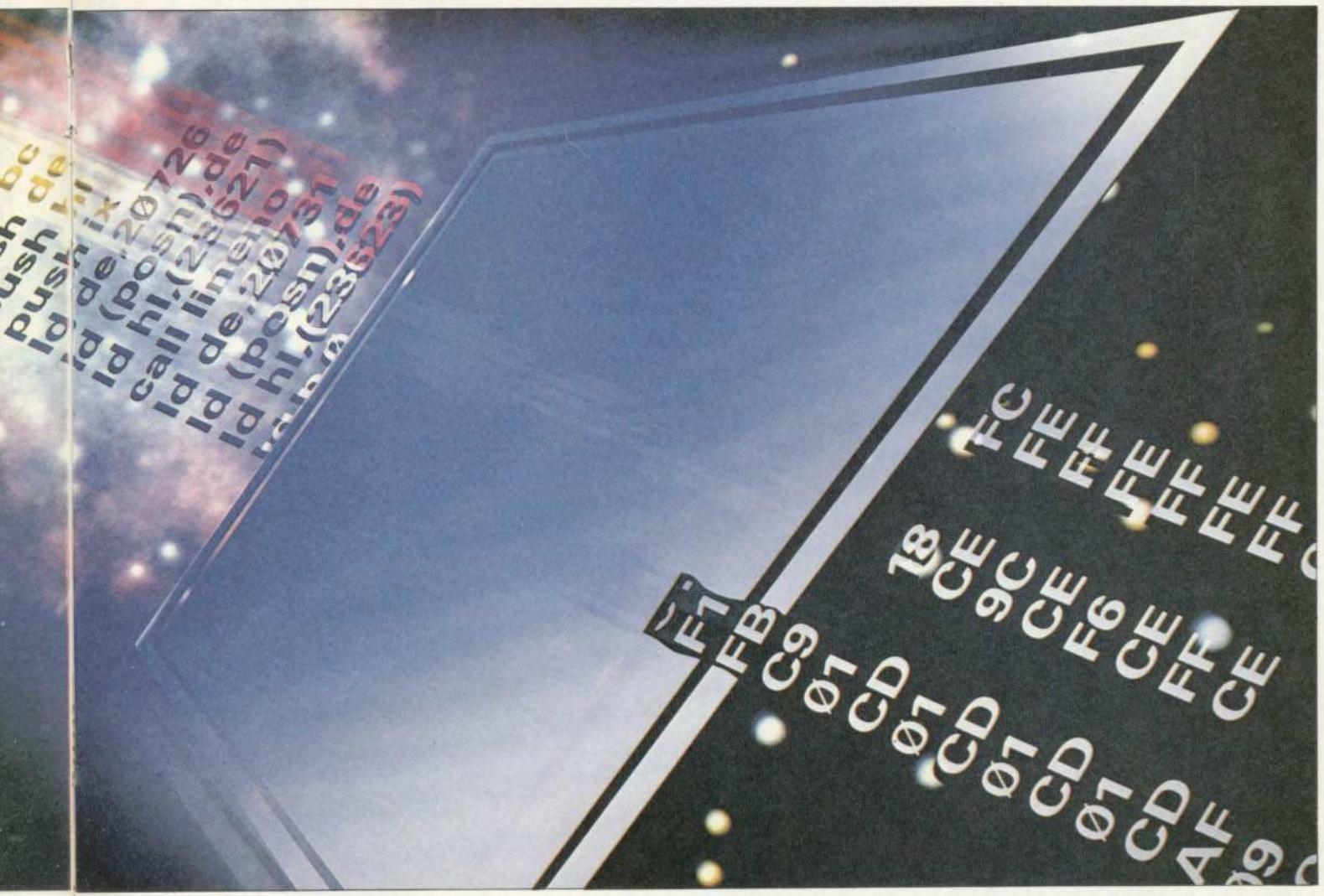
```
R i=1 TO 8/j: READ s$(i,j): NEX
T i: NEXT j: FOR j=1 TO 18: REA
D t(j),u$(j): NEXT j
5060 DATA "b","c","d","e","h",
"1","(h1)","a","bc","de","h1","s
p",235,"de",8,"af",227,"(sp)",6
0742,"0",60758,"1",60766,"2",23
3,"(h1)",56809,"(ix)",65001,"(i
y)",10,"(bc)",26,"(de)",60767,
"r",2,"(bc)",18,"(de)",60751,"r"
,249,"sp",60743,"i",60759,"i"
5070 DEF FN b(x,i)=INT (x/b(i+1
))-INT (x/b(i+2))*2
5080 DEF FN x(x,i)=x-b(i+2)*FN
b(x,i)+b(i+1)
5090 DEF FN j(x,i)=INT x-b(i+1
)*INT (x/b(i+1))
5100 DEF FN e(i$,j$)=(i$=j$( TO
LEN (i$)))
5110 FOR i=1 TO 110: READ k$(i)
,k(i),m(i): IF NOT FN e("(*",k$(i
)) THEN NEXT i
5120 DATA "1d",10,10,"1d",26,10
,"1d",60767,10,"1d",60759,10,"1
d",2,138,"1d",18,138,"1d",60751
,138,"1d",60743,138,"1d",64,6,"1
d",50,202,"1d",58,74,"1d",249,
139,"1d",34,195,"1d",42,67,"1d"
,60779,197,"1d",60771,199,"1d",
97,165,"1d",64,54
5130 DATA "adc",136,50,"adc",60
746,3,"add",128,50,"add",9,149,
"and",160,48,"or",176,48,"xor",
168,48,"nop",0,0,"sub",144,48,"
sbc",152,50,"sbc",60738,3,"cp",
184,48,"jp",130,45,"jp",233,9,"
jp",56809,9,"jp",65001,9,"jp",1
31,49,"jr",96,45,"jr",88,41
5140 DATA "call",132,45,"call",
141,41,"ret",201,0,"djnz",74,40
"dec",11,17,"dec",5,16,"inc",3
,17,"inc",4,16,"push",197,17,"p
op",193,17,"di",243,0,"ei",251,
0,"halt",118,0,"ex",235,139,"ex
",8,15,"ex",227,143,"exx",217,0
5150 DATA "rst",199,132,"rts",1
92,5,"bit",52032,20,"defb",-256
,40,"ccf",63,0,"scf",55,0,"cpl"
,47,0,"cpd",60841,0,"cpdr",6085
7,0,"cpi",60833,0,"cpir",60849,
0,"daa",39,0,"im",60742,8,"im",
60758,8,"im",60766,8
5160 DATA "in",60736,130,"in",1
49,42,"ind",60848,0,"indr",6081
0,0,"ini",60840,0,"ldd",60840,0
,"laddr",60856,0,"ldi",60832,0,"
ldir",60848,0,"neg",60740,0,"ot
dr",60859,0,"otir",60851,0,"out
",60737,2,"out",141,170,"outd",
60843,0,"outi",60835,0
5170 DATA "res",52096,20,"reti"
,60749,0,"retn",60741,0,"rl",51
```

Um programa que calcula a dimensão dos saltos e faz a tradução dos códigos mnemônicos Assembly para o sistema hexadecimal: o que mais poderíamos desejar na vida?



```
984,64,"rla",23,0,"rlc",51968,1
6,"rlca",7,0,"rld",60783,0,"rr"
,51992,64,"rra",31,0,"rrc",5197
6,16,"rrca",15,0,"rrd",60775,0
5180 DATA "set",52160,20,"sla",
52000,16,"sra",52008,16,"srl",5
2024,16,"defw",-256,41
5190 DATA "/*",0,0: LET ii=i: LE
T k(110)-ii
5200 LET b=0: LET ba=PEEK 23635
+256*PEEK 23636+4: LET n=1
5210 LET cc=1: IF PEEK ba<>234
THEN LET n=n-1: GOTO 5250
5220 LET cc=cc+1: LET ba=ba+1:
IF PEEK ba=13 THEN LET ba=ba+5
: LET n=n+1: GOTO 5210
5230 LET t$(n,cc)=CHR$ PEEK ba
```

- TRADUÇÃO DE ASSEMBLY PARA CÓDIGO DE MÁQUINA
  - COMO CALCULAR SALTOS E DESVIOS
  - O USO DE RÓTULOS
  - RESERVE ESPAÇO PARA DADOS E VARIÁVEIS
  - COMO COLOCAR OS CÓDIGOS NA MEMÓRIA USANDO POKE
  - TESTE O PROGRAMA



```

5240 GOTO 5220
5250 FOR g=1 TO 100: LET r(g)=g
-1: NEXT g: LET fh=100
5300 LET k0=0: LET k9=99: LET p
0=0: LET vv=0
5310 LET k=k0: LET p=p0
5320 GOSUB 8000
5330 GOSUB 7000: LET o$=i$: IF
o$(1)="*" THEN PRINT o$;: GOTO
5320
5340 IF o$="end" THEN PRINT ''
    endereço final ";p-1
5350 IF o$="end" THEN LET p0=p
: GOTO 9999
5370 IF o$<>"org" THEN GOTO 54
00
5380 GOSUB 7000: LET s=0: IF i$
```

```

(1)=** THEN LET s=p: LET i$=i
$(2 TO )
5390 LET p=VAL i$+s: PRINT ""
    org ";p;: GOTO 5320
5400 IF p=0 THEN PRINT "(falta
    org)": LET p=50000
5410 LET p$=o$+"!": FOR i=1+18*
(o$<>"1d") TO 110: IF o$<-k$(i)
    AND p$>k$(i) THEN GOTO 5500
5420 NEXT i: PRINT o$
5430 IF i$(1)=".." THEN LET i$=
i$(2 TO )
5440 GOSUB 9000: LET gg=r(g)
5450 IF gg<=100 THEN LET s=SGN
    z(gg): LET b=INT (ABS z(gg)/65
536): LET r=ABS z(gg)-b*65536:
    LET q=PEEK r+256*PEEK (r+1): PO
KE r, FN j(p*s+q,8): PRINT " POK
E ";r;" COM ";FN j(p*s+q,8): IF
    b THEN POKE (r+1),FN j((p*s+q
)/256,8): PRINT " POKE ";r+1;" COM ";
    FN j((p*s+q)/256,8)
5460 IF gg<=100 THEN LET gh=r(
gg): LET r(gg)=fh: LET fh=gg: L
ET gg=gh: GOTO 5450
5470 IF i$="" THEN LET r(g)=p+
100: GOTO 5330
5480 PRINT " (Linha nao foi rec
onhecida)"
5490 GOTO 5420
5500 LET z=0: LET r=0: LET e=0:
    PRINT " ";o$;
5510 LET op=k(i): IF m(i)=0 THE
    N GOTO 6090

```

```

5520 GOSUB 7000: LET a$=i$: PRI
NT " ";a$;
5530 LET m=m(i): LET op=k(i): L
ET b=FN b(m,0): LET b7=b+2*FN b
(m,7)+1: LET z=0: IF FN j(m,3)<
2 THEN LET c$=a$: GOTO 5720
5540 FOR j=1 TO LEN a$: IF a$(j)
)="-," THEN GOTO 5580
5550 NEXT j: IF o$="rst" OR o$=
"rts" THEN GOTO 5580
5560 IF FN e(o$,k$(i+1)) THEN
LET i=i+1: GOTO 5530
5570 PRINT "(deve haver dois o
perandos)": GOTO 5320
5580 LET b$=a$( TO j-1): LET c$
=a$(j+1 TO )
5590 IF FN b(m,2) THEN GOTO 56
5600
5600 IF FN b(m,7) THEN LET d$-
c$: LET c$=b$: LET b$=d$
5610 IF b$="ahl"(b+1 TO b+b+1)
THEN GOTO 5720
5620 IF b$="(c)" AND (o$="in" O
R o$="out") THEN GOTO 5720
5630 IF (FN e(o$,k$(i+1))) AND
(FN j(m(i+1),3)>-2) THEN LET i
=i+1: GOTO 5530
5640 PRINT "(primeiro operando
deve ser a ou hl)": GOTO 5320
5650 IF FN b(m,1) THEN GOTO 56
90
5660 LET e$=(b$+" ")( TO 4):
FOR j=1 TO 8: IF e$=r$(j,b7) TH
EN LET op=op+8*(j-1)*(b7<4)+16
*(j-6)*(b7-4)*(j>3): LET z=(j-1
)*(b7-4)*(j<-3): GOTO 5710
5670 NEXT j: IF p$>k$(i+1) AND
(FN j(m(i+1),3)>-2) THEN LET i
=i+1: GOTO 5530
5680 PRINT "(primeiro operando
deve ser bit ou sinalizador)": GOTO 5320
5690 IF FN b(m,7) THEN LET d$-
c$: LET c$=b$: LET b$=d$: GOTO 5660
5700 LET x=8: GOSUB 5750: IF e
THEN GOTO 5730
5710 IF c$="-" THEN GOTO 6090
5720 LET x=1+15*b+7*(op<=6 AND
op>=4 OR b$="(c)": LET b$=c$:
GOSUB 5750: IF NOT e THEN GOTO
6090
5730 IF e=2 OR p$>k$(i+1) AND F
N j(m(i+1),3)=FN j(FN x(m,0),3)
THEN LET e=0: LET i=i+1: GOTO
5530
5740 GOTO 5320
5750 LET r=0: IF FN b(m,4) AND
FN e("(("( TO NOT b),b$) THEN L
ET z2=FN e("ix",b$(2-b TO ))+" "
)+2*FN e("iy",b$(2-b TO ))+" "):
IF z2 THEN LET z=z2: LET e$=b
$( TO LEN b$-NOT b): LET b$="(h
1)"(1+b TO 4-b): LET f$="0"+e$(
4-b TO )
5760 IF FN b(m,3) THEN GOTO 57
90
5770 LET e$=(b$+" ")( TO 4):
FOR j=1 TO 8/(b+1): IF e$=s$(j,
b+1) THEN LET op=op+(j-1)*x: R
ETURN
5780 GOTO 5810
5790 LET j2=9+9*(o$="ld"): FOR

```

```

j=j2-8 TO j2: IF k(i)<>t(j) THE
N GOTO 5810
5800 IF FN e(b$,u$(j)) THEN RE
TURN
5810 NEXT j: IF b$="af" THEN I
F FN e("p",o$) THEN LET op=op+
48: RETURN
5820 IF FN b(m,6) AND FN e("(",b$)
THEN LET b$=b$(2 TO LEN b$-
1): GOTO 5860
5830 IF FN b(m,5) THEN LET op=
FN x(op+6*NOT b,6): GOTO 5860
5840 IF p$>k$(i+1) THEN LET e=
2: RETURN
5850 PRINT "(operando incompati
vel)": LET e=1: RETURN
5860 LET r=65536
5870 LET s=1
5880 IF b$="" THEN GOTO 6080
5890 LET x$=b$(1): LET d$=b$(2
TO ): IF x$=="*" THEN LET r=r+p
*s: LET b$=d$: GOTO 5870
5900 IF x$=="+" THEN LET b$=d$:
GOTO 5880
5910 IF x$=="-" THEN LET b$=d$:
LET s=-s: GOTO 5880
5920 IF x$="**" THEN LET r=r+CO
DE d*$*s: LET b$=d$(2 TO ): GOTO
5870
5930 LET q=0: IF x$<>"%" OR d$<
"0" OR d$>"2" THEN GOTO 5960
5940 IF d$>="0" AND d$<"2" THEN
LET q=q*2+CODE d$-48: LET d$=
d$(2 TO ): GOTO 5940
5950 LET r=r+q*s: LET b$=d$: GO
TO 5870
5960 IF x$<>"$" OR d$<"0" OR d$>
="q" THEN GOTO 6000
5970 LET x$=CHR$ CODE d$: FOR g
=0 TO 15: IF x$<>h$(g+1) AND x$>
>g$(g+1) THEN GOTO 5990
5980 LET q=q*16+g: LET d$=d$(2
TO ): GOTO 5970
5990 NEXT g: LET r=r+q*s: LET b
$=d$: GOTO 5870
6000 IF x$<"a" OR x$>"z" THEN
GOTO 6040
6010 LET i$=b$: GOSUB 9000: IF
i$<>"" THEN GOSUB 9400
6020 IF r(g)<>23000 AND r(g)>10
0 THEN LET r=r+(r(g)-100)*s: L
ET b$=i$: GOTO 5870
6030 IF r(g)=23000 OR r(g)<=100
THEN LET gh=r(fh): LET r(fh)=
r(g): LET r(g)=fh: LET fh=gh: L
ET z(r(g))=(p+SGN op+(ABS op>25
5)+2*(z>0)+65536*((b OR FN b(m,
6)) AND o$<>"jr"))*s: LET b$=i
: GOTO 5870
6040 IF x$<"0" OR x$>"9" THEN
LET r=0: GOTO 6070
6050 IF b$>="0" AND b$<":" THEN
LET q=q*10+CODE b$-48: LET b$-
b$(2 TO ): GOTO 6050
6060 LET r=r+s*q: GOTO 5870
6070 PRINT "(enderecamento inco

```



```

7020 LET k9=k1: IF t$(k,k1)="" "
THEN GOTO 7010
7030 IF k9>LEN t$(k) THEN LET
i$=t$(k)(k1 TO ): RETURN
7040 IF t$(k,k9)<>" " THEN LET
k9=k9+1: GOTO 7030
7050 LET i$=t$(k)(k1 TO k9-1):
RETURN
8000 IF k>0 THEN IF t$(k)(k9 T
O )>t$(99) THEN PRINT t$(k)(k9
TO );
8010 POKE 23692,0: LET k=k+1: L
ET k9=0
8020 PRINT : RETURN
9000 LET x$=""
9010 IF i$<"a" OR i$>"z" THEN
GOTO 9030
9020 LET x$=x$+i$(1): LET i$=i$(
2 TO ): GOTO 9010
9030 IF i$<>"" THEN RETURN
9400 FOR g=1 TO vv: IF FN e(x$,
z$(g)) THEN RETURN
9410 NEXT g: LET vv=vv+1: LET z
$(vv)=x$: LET g=vv: LET r(g)=23
000
9420 RETURN

```

### COMO FUNCIONA O PROGRAMA

Começando na linha 5000, o programa deixa espaço para seu programa em Assembly, que deve ser colocado em linhas **REM**. Cada instrução deve ser escrita com letras minúsculas e posicionada em uma linha **REM** separada.

Antes de montar o programa, devemos proteger uma área de memória para que o Assembler possa colocar os códigos, usando **CLEAR**. A primeira linha de seu programa deve ser mais ou menos assim:

```
10 REM org 55000
```

32000 é o endereço inicial do programa em código. Obviamente, ele deve estar na área protegida da memória.

Se esquecermos de definir **org**, ou seja, a origem, o Assembler colocará o programa a partir de 50000.

O programa aceita os códigos mnemônicos-padrão do chip Zilog Z-80, com exceção dos comandos de retorno condicional. Entretanto, o comando de retorno incondicional de sub-rotinas, cujo mnemônico é **ret**, funciona em nosso Assembler.

Algumas vezes, contudo, utilizamos retornos condicionais, como **ret nz**, que significa: "retorne se não for zero". Neste programa, porém, digite a instrução **rts nz**. A razão disso é que **rts** deve substituir **ret** sempre que forem necessários retornos condicionais. Quando se tratar de retorno incondicional, ou seja, quando **ret** não for seguido de nenhuma letra, use o mnemônico padrão **ret**.

Números hexadecimais devem ser



O que acontecerá se houver um erro em meu programa fonte?

Nosso Assembler é capaz de emitir mensagens de erro, pois a sua estrutura de programação está preparada para detectar falhas no programa em Assembly. Alguns comandos, por exemplo, só trabalham com os registros **a** e **h1**. Se tentarmos usá-los com outros registros, o Assembler dirá: "Primeiro operando deve ser a ou h1"

Se um comando não for reconhecido devido a um erro de digitação, seremos automaticamente informados: "Linha não foi reconhecida". A expressão "Deve haver dois operandos" significa que deixamos de digitar um número vital após o comando. "Operando incompatível" indica um uso inadequado do comando. "Primeiro operando deve ser bit ou sinalizador" significa que um operando inadequado foi usado em um comando de desvio ou de atribuição de bits.

Existe uma maneira de se programar em Assembler nos ZX-81?

Existe; mas o meio para isso não é um programa montador desenvolvido em BASIC, como os que serão apresentados mais adiante para as linhas MSX, Apple II, TRS-80 e TRS-Color.

Montadores em BASIC para o ZX-81 (com seus similares nacionais TK-85, CP-200, etc.) são inviáveis porque o programa resultante seria muito grande e de difícil digitação e operação: na verdade, ele não caberia na memória da maioria dos micros dessa linha; além disso, o processo de tradução e montagem desse programa seria extremamente lento.

Para comprovar isso, observe a listagem para o ZX Spectrum que acompanha este artigo: é a inexistência de declarações **READ** e **DATA** no ZX-81, o que dificulta a programação.

Seria possível programar um Assembler reduzido, sem rótulos alfabéticos, e com um conjunto menor de comandos. Mas isso o tornaria inútil para quem quer aprender a programar seriamente, e impossibilitaria o usuário de digitar e testar os programas em código Assembler que aparecerão em lições futuras. Por isso, quem quiser programar em Assembler no ZX-81, deve recorrer a um programa tradutor compacto, eficiente e rápido, escrito em código de máquina e disponível comercialmente.

```

rreto)"
6080 LET r=r-(p+2)*(o$="djnz" O
R o$="jr"): RETURN
6090 PRINT TAB 16,: LET by=p/25
6: GOSUB 6190: LET by=p: GOSUB
6190: GOSUB 6160
6100 IF z THEN LET by=189+z*32
: GOSUB 6180: GOSUB 6160
6110 IF op>=0 THEN LET by=op/2
56: GOSUB 6170: GOSUB 6150: LET
by=op: GOSUB 6180
6120 IF r=0 THEN GOTO 5320
6130 GOSUB 6160: LET by=r: GOSU
B 6180: IF (b OR FN b(m,6)) AND
o$<>"jr" THEN LET by=r/256: G
OSUB 6180
6140 GOTO 5320
6150 IF z AND INT by AND NOT b
THEN GOSUB 6260: LET by=VAL f$ :
GOSUB 6180: LET z=0
6160 PRINT " ";: RETURN
6170 IF INT by<0 THEN RETURN
6180 LET by=FN j(by,8): POKE p,
by: LET p=p+1
6190 LET by=FN j(by,8): PRINT h
$(1+INT (by/16));h$(FN j(by,4)+
1);
6200 RETURN
7000 IF k>n THEN LET i$="end":
RETURN
7010 LET k1=k9+1: IF k9>=LEN t$(
k) THEN LET i$="/faltando/":
RETURN

```

precedidos do caractere \$; números binários, do caractere %. Qualquer número que não seja precedido de algum caractere será interpretado como decimal. Qualquer palavra que não seja um comando Assembly será considerada como um rótulo (label). Evite usar palavras parecidas ou muito longas; números não são permitidos.

Para que o Assembler saiba onde parar, termine seu programa com **REM end**.

Uma vez colocado o programa Assembly a ser montado (o *programa fonte*) nas linhas **REM** iniciais, basta rodar o Assembler para se obter uma listagem dos códigos equivalentes na tela: o *programa objeto*. O Assembler coloca o programa objeto simultaneamente na memória e na tela.

Se a esta altura for cometido algum erro na digitação das linhas, será suficiente, para corrigi-lo, listar o Assembler, editar a linha **REM** correspondente e usar novamente **RUN**.

Montado o programa, o endereço final da rotina em código de máquina aparecerá na tela.

Para executar o programa em código usamos instruções do tipo **RANDOM USR**.

Se quisermos gravar o programa objeto, devemos digitar:

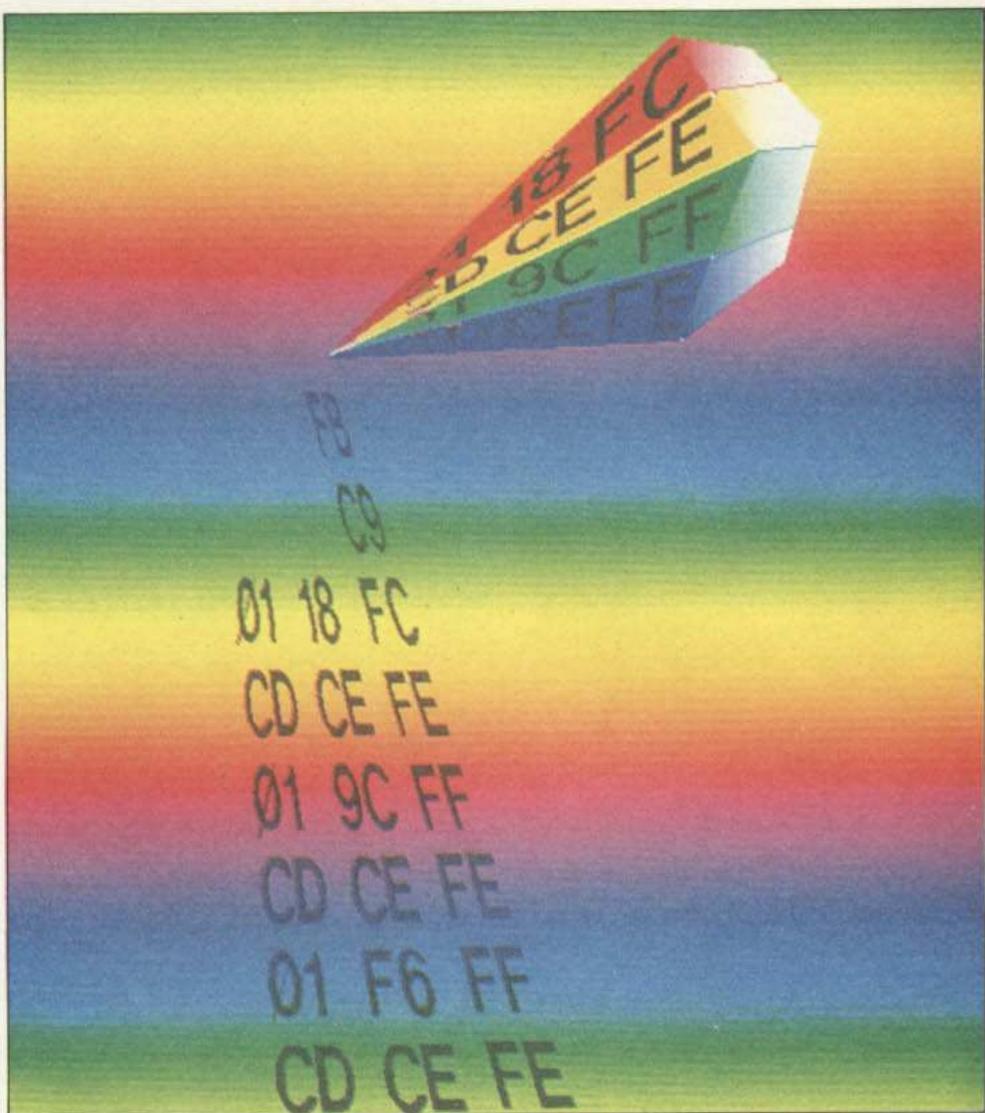
**SAVE "nome" CODE** endereço inicial, número de bytes.

## MICRO DICAS

### COMO DETECTAR ERROS EM PROGRAMAS LONGOS

Sempre que a mensagem "**OUT OF DATA**" aparecer na tela, verifique se está faltando alguma coisa nas linhas **DATA** (lembre-se de que a ausência de uma simples vírgula pode deitar a perder todo o programa). Um dos erros mais comuns na digitação de longos programas consiste justamente na omissão de trechos de linhas **DATA**. Por isso, caso seu Assembler não funcione ao ser rodado pela primeira vez, não se desespere: ele certamente "empacou" devido a erros de digitação.

Esses erros podem ser detectados por meio de um programa de rastreamento, que escreve na tela o número da linha BASIC que está sendo executada. Um programa de rastreamento completo será apresentado mais adiante.



"Nome" é o nome do programa objeto e deve estar entre aspas. **CODE** informa ao computador que está sendo gravado um programa em código de máquina e não em BASIC. O endereço inicial é a origem do programa na memória. Podemos calcular o número de bytes subtraindo a origem do endereço final e somando 1.

O Assembler e o programa fonte podem ser gravados juntos, como um programa BASIC, usando-se **SAVE** da maneira habitual.

### UM TESTE

Para testar seu programa, entre o programa de deslocamento horizontal da tela para a direita, que se encontra na página 215. Uma vez traduzido para código, esse programa ficará assim:

```
11 FF 57 21 FE 57 06 C0 C5 1A  
01 1F 00 ED B8 12 2B 1B C1 10  
F3 C9
```

Note que devemos usar letras minúsculas para digitar o programa fonte. Nossa Assembler não reconhece comandos em maiúsculas.

### S

Não apresentaremos um Assembler para o ZX-81, pois é muito difícil escrever um programa desse tipo em BASIC para esse micro. Se você tem um ZX-81 e está interessado em código de máquina, sugerimos a compra de um Assembler gravado em fita cassete e disponível em lojas especializadas.

Caso já disponha de algum, teste o programa de deslocamento da tela para a direita apresentado na página 217 que, uma vez traduzido para código de máquina, ficará assim:

```
2A 0C 40 11 16 03 19 54 5D 13  
06 18 C5 1A 01 1F 00 ED B8 12  
2B 1B 1B C1 10 F1 C9
```

# UM PROFESSOR DE DATILOGRAFIA

Aspiração de todo programador que se preza, a eficiência no processamento de textos e na digitação de programas tem um pré-requisito fundamental: saber datilografia.

O futuro nos reserva, certamente, muitas surpresas no campo das relações entre o homem e a máquina. Uma delas pode ser a abertura de novos canais de comunicação entre o computador e o usuário — como, por exemplo, a voz humana. Atualmente, alguns desses sistemas de controle mais sofisticados já vêm sendo aplicados em máquinas industriais e comerciais a um preço relativamente baixo. A maioria dos usuários, porém, vai ter que esperar muito tempo até ter acesso a essas inovações. Enquanto isso, o velho teclado continuará a ser o dispositivo básico de entrada de um computador.

Na verdade, o teclado é um instrumento de eficiência razoável para se transmitir instruções ao computador.

Sua utilização, no entanto, consome muito tempo, especialmente quando o operador não é um exímio datilografo. Essa perda de tempo é visível quando tentamos copiar uma listagem e somos obrigados a olhar do teclado para a tela e desta para o papel, só para nos assegurarmos de que os dados estão sendo introduzidos corretamente.

Assim, por que não utilizar o computador no aprimoramento do modo de datilografar? Fazendo isso, você economizará tempo e evitará dores de cabeça, quando quiser escrever programas ou explorar o potencial de um computador como processador de textos. O programa que se segue ajudará a se familiarizar com o teclado e aumentar a velocidade e a precisão na datilografia.

O Sinclair Spectrum apresenta a vantagem de ter um teclado de difícil manipulação e grande velocidade. A introdução de programas, contudo, exige apenas um toque de tecla para cada comando em BASIC. Note que não estamos apresentando uma versão do programa para o Sinclair ZX-81. Isso se deve ao fato de que sua execução se tor-

- PARA QUE APRENDER DATILOGRAFIA?
- AS TECLAS DE APOIO
- MELHORE A VELOCIDADE E A PRECISÃO NA DATILOGRAFIA

naria muito lenta, perdendo eficácia como forma de treinamento.

## O PROGRAMA

Do mesmo modo que qualquer curso padrão de datilografia, nosso programa tem por objetivo desenvolver metodicamente a destreza do iniciante. Esse programa, no entanto, não é nenhum processo mágico e, como qualquer outra coisa que se queira aprender, exige dedicação.

Para efeitos didáticos, ele foi dividido em etapas mais ou menos fáceis, com seções extras de programação acrescentadas a cada uma das fases. Dessa maneira, é possível avançar através de missões cada vez mais difíceis. Outra virtude de nosso curso consiste em apresentar, ao longo de seu desenvolvimento, uma atualização constante da velocidade e da precisão atingidas pelo iniciante.

Se você é usuário de um Apple II, ficará sem o recurso de cronometragem, devido à falta de um "clock" acessível nesse micro. Talvez a cronometragem

 commodore

64



SELECIONE A TECLA INDICADA  
AO ASTERISCO

A S D F G H J K L ;

LEVEL 2 ...

SELEÇÃO NIVEL  
<1-5>

TEMPO = 18.02 SEGUNDOS  
NÚMERO DE ERROS = 0

manual o auxílio, nos níveis 4 e 5, a controlar seus progressos na datilografia.

#### CONHEÇA AS TECLAS DE APOIO

Ser um exímio datilógrafo significa ser capaz de pressionar as teclas corretas sem precisar olhar para elas sempre que se quer localizá-las. Evidentemente, isso só poderá ser feito quando se sabe previamente onde se encontram as letras no teclado. É preciso, portanto, "educar" os dedos para que eles encontrem automaticamente as teclas desejadas. O primeiro passo para isso consiste em posicionar os dedos sempre na mesma posição sobre o teclado. Datilógrafos profissionais fazem isso assegurando-se de que os dedos das duas mãos repousem sempre sobre as mesmas teclas, as denominadas *teclas de apoio*. Estas são as teclas da fileira do meio do

teclado, começando pelas letras A, S, D, F... Portanto, estas serão as teclas que devemos memorizar em primeiro lugar.

Uma vez dominada a localização dessas teclas básicas, o restante do teclado pode ser alcançado movendo-se cada dedo para cima ou para baixo.

Para obter o máximo rendimento no curso será preciso ater-se a essa disposição, sem nunca tentar "enganar" o computador. Por outro lado, os que insistem em continuar "catando milho" devem também procurar obter um gradativo aprimoramento de sua destreza.

Ao introduzirmos o programa — tarefa que deverá tornar-se cada vez mais fácil à medida que o curso prosseguir — devemos ficar com as mãos pousadas sobre o teclado, aguardando as instruções que aparecerem na tela e que serão explicadas a seguir.

O dedo mínimo da mão esquerda deve repousar sobre a tecla A, o anular so-

bre o S, assim por diante, terminando com o dedo indicador sobre a tecla F. O indicador direito, por sua vez, deve estar na tecla J, o dedo médio na tecla K, e assim por diante. Os polegares (exceto no caso do Spectrum) devem posicionar-se sobre a barra de espaçoamento. As letras G e H, portanto, não são cobertas por nenhum dedo. Quando é necessário acioná-las, o G pode ser alcançado com o dedo indicador esquerdo, e o H com o indicador direito.

Parece complicado, mas não se preocupe: a primeira parte do programa se destina justamente a familiarizá-lo com as letras do teclado.

#### COMO FUNCIONA O PROGRAMA

Ao se executar o programa, a tela mostrará cinco opções, que correspon-

dem a níveis de dificuldade crescente. Você deve escolher uma entre elas. Se você está começando a aprender, precisa estudar primeiro a lição 1 para, só depois de dominar seu conteúdo, passar para a lição 2, e assim por diante.

dor de cada mão), voltando-se às teclas de apoio logo em seguida. Utilize o dedo mínimo direito para as teclas ; e Ç.

O asterisco continuará no mesmo lugar, caso seja cometido algum engano. Uma tecla batida corretamente será anunciada por um bip, e por um som grave se houver um erro ou se uma letra for omitida. (No Apple ocorrerá o oposto.)

Ao fim de cada exercício, a tecla mostrará o tempo gasto e o número de erros cometidos.

### NÍVEL 3

Este nível também apresenta letras aleatoriamente, uma por vez. Agora, porém, a tela não indicará sua posição no teclado. Em vez disso, elas serão mostradas individualmente no centro da tela. Mais uma vez, as letras serão em número de vinte.

### NÍVEL 4

Finalmente, pode-se datilografar algumas palavras. As expressões deste



### NÍVEL 1

A tela apresentará as teclas de apoio em uma linha, na mesma ordem em que estão no teclado. Ao mesmo tempo, será mostrado um asterisco que, deslocando-se acima das letras, percorrerá o teclado do A até o L no Spectrum e no CP400, até o ; (ponto e vírgula) no Apple, até o Ç no MSX. Para se acionar as teclas, deve-se tocá-las na sequência indicada pelo asterisco e com o dedo correto (lembre de que o G e o H exigem um deslocamento do dedo indica-

### NÍVEL 2

Uma vez familiarizado com a posição das letras no teclado, você pode passar para este nível, que é semelhante ao primeiro, exceto que agora o asterisco se deslocará aleatoriamente sobre as letras (não vale trapacear olhando a posição da letra do teclado). Novamente, o tempo que se levar para completar o exercício (serão apresentadas vinte letras aleatoriamente) e o número de erros cometidos serão apresentados no final.

exercício utilizam apenas as letras das teclas de apoio: FADA, por exemplo. Ao contrário do nível anterior, serão apresentadas no centro da tela vinte palavras (e não letras) aleatoriamente e em sucessão, isto é, uma após a outra. À medida que as palavras forem sendo copiadas, um asterisco indicará cada uma de suas letras. E, mais uma vez, será apresentado o resultado do tempo gasto e da precisão. Não se deve trapacear olhando para o teclado; senão, não será pos-

sível completar os testes mais difíceis, apresentados nas próximas lições.

### NÍVEL 5

Desta vez será escrita uma sentença com cerca de seis palavras aleatórias. À medida que estas forem sendo digitadas, as letras aparecerão uma após a outra. Quando a linha toda tiver sido datilografada, será mostrado o desempenho do datilógrafo, assim como um valor correspondente à velocidade em termos de palavra por minuto. Ao se digitar as palavras que surgirem na tela, não se deve esquecer dos espaços em branco entre elas; esse espaço é obtido quando se toca na barra de espaçojamento com um dos polegares. No Spectrum será preciso pressionar com o dedo mínimo a barra de espaços, situada numa posição um tanto incômoda.

Deve-se exercitar cada um dos níveis,

até que se torne fácil executá-los. Nos próximos capítulos acrescentaremos letras e palavras novas.

### S

```

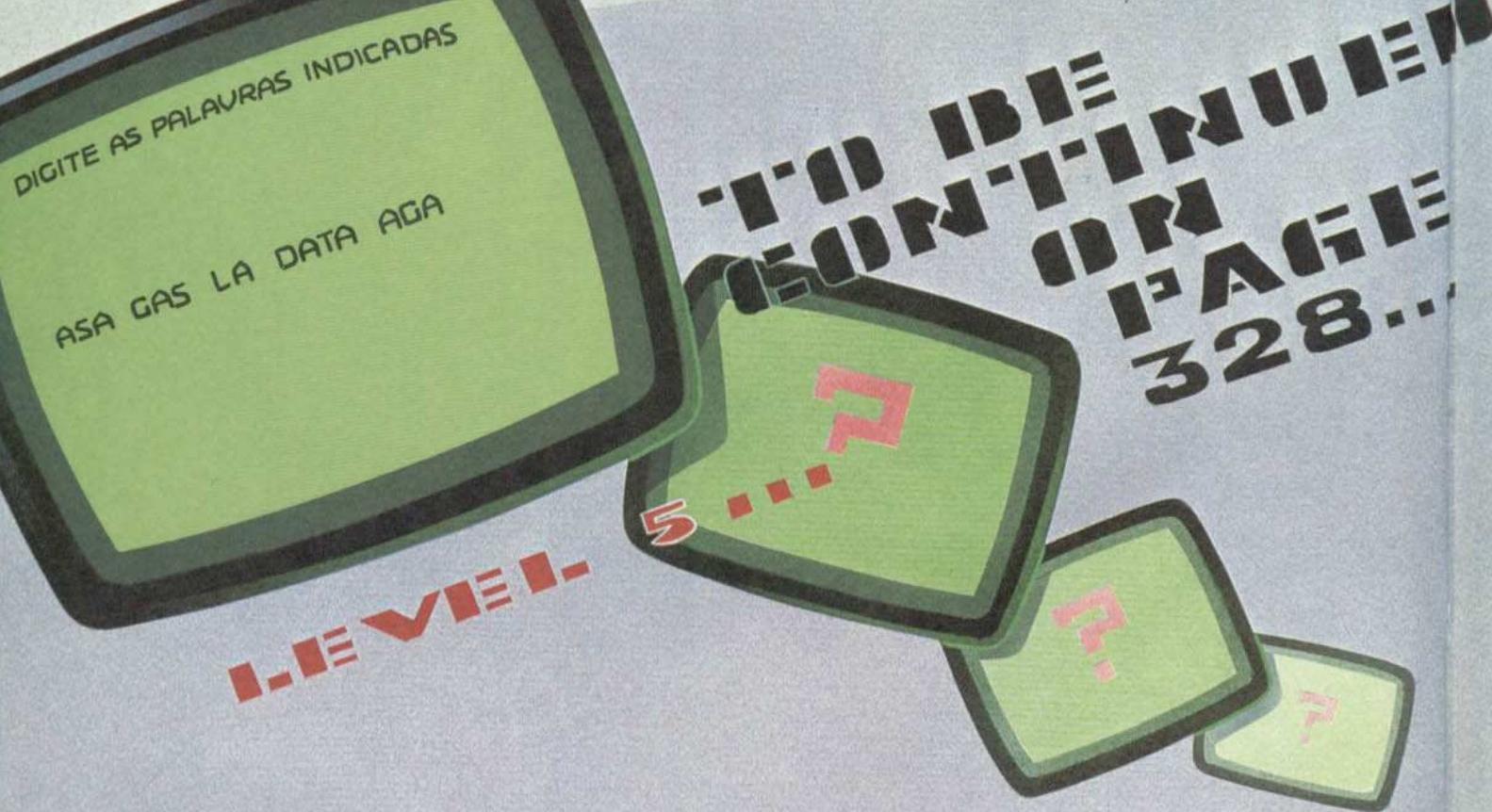
10 BORDER 1: PAPER 1: INK 7:
CLS
20 POKE 23658,8: LET ER=0
30 LET S$="ASDFGHJKL"
100 PRINT INVERSE 1;AT 8,3;"PROFESSOR DE DATILOGRAFIA"
110 PRINT "'TAB 6;"SELEÇÃO N
IVEL (1 A 5)"
120 IF INKEY$="" THEN GOTO 120
130 LET AS=INKEY$: IF AS<"1"
OR AS>"5" THEN GOTO 120
140 SOUND .2,10
150 GOSUB VAL AS*100+100
160 GOTO 20
200 GOSUB 1000
210 FOR K=7 TO 23 STEP 2
220 PRINT AT 10,K;"*"
230 LET RS=S$(K-5)/2

```

```

240 GOSUB 1100
250 IF C=0 THEN GOTO 240
260 PRINT AT 10,RN;" "
270 NEXT K
280 CLS : GOTO 1300
300 GOSUB 1000
310 FOR K=1 TO 20
320 LET RN=INT (RND*9)*2+1
330 PRINT AT 10,RN+6;"*": LET
RS=S$((RN+1)/2)
340 GOSUB 1100: IF C=0 THEN
GOTO 340
350 PRINT AT 10,RN+6;" "
360 NEXT K
370 CLS : GOTO 1300
400 CLS : PRINT "DIGITE A LETR
A INDICADA NA TELA"
410 FOR N=1 TO 100: NEXT N
420 POKE 23672,0: POKE 23673,0
430 FOR K=1 TO 20
440 LET RN=INT (RND*9)+1
450 PRINT AT 11,16;S$(RN)
460 LET R$=S$(RN)
470 GOSUB 1100: IF C=0 THEN
GOTO 470
480 PRINT INVERSE 1;AT 11,16;
" ": NEXT K
490 CLS : GOTO 1300

```



```

500 CLS : PRINT "DIGITE A PALA  
VRA INDICADA": POKE 23672,0:  
POKE 23673,0  
510 LET TL=0: FOR N=1 TO 20:  
RESTORE : LET RN=INT (RND*24)+  
1  
520 FOR K=1 TO RN: READ TS:  
NEXT K  
530 PRINT AT 10,13;"      ":"  
PRINT AT 10,13;TS  
540 FOR M=1 TO LEN TS: PRINT  
AT 9,11+M;" *   "  
550 LET RS=TS(M): GOSUB 1100  
560 IF C=0 THEN GOTO 550  
570 NEXT M  
580 LET TL=TL+LEN TS: NEXT N  
590 CLS : PRINT AT 18,0;"PALAV  
RAS POR MINUTO= ";INT (TL*500/  
(PEEK 23672+256*PEEK 23673)+.5  
): GOTO 1300  
600 CLS : PRINT "DIGITE AS PAL  
AVRAS INDICADAS": LET TS=""  
610 FOR N=1 TO 6: RESTORE :  
LET RN=INT (RND*24)+1: FOR K=1  
TO RN: READ XS: NEXT K  
620 LET TS=TS+XS+" "  
630 NEXT N: LET TS=TS( TO LEN  
TS-1)  
640 POKE 23672,0: POKE 23673,0  
: PRINT AT 10,0;TS  
650 PRINT AT 12,0;: FOR M=1 TO  
LEN TS  
660 LET RS=TS(M): GOSUB 1100  
670 IF C=0 THEN GOTO 660  
680 PRINT TS(M):: NEXT M  
690 LET TL=LEN TS: GOTO 590  
1000 CLS : PRINT "PRESSIONE A T  
ECLA INDICADA PELO ASTERISCO"  
1010 PRINT AT 12,7;"A S D F G H  
J K L"  
1020 FOR K=1 TO 100: NEXT K: PO  
KE 23672,0: POKE 23673,0  
1030 RETURN  
1100 PAUSE 0  
1110 LET AS=INKEYS: IF AS<>RS T  
HEN SOUND .2,-10: LET ER=ER+1:  
LET C=0: RETURN  
1120 SOUND .05,20: LET C=1: RET  
URN  
1300 PRINT AT 19,0;"TEMPO= ";(P  
EEK 23672+256*PEEK 23673)/50;"  
SEGUNDOS": PRINT "NUMERO DE ER  
ROS= ";ER  
1310 PAUSE 100: RETURN  
2000 DATA "FADA","GALA","SALA",  
"DADA","ASA","LAKA","ALAGA","AL  
ADA","LA","GAS","DALLAS","SAGA"  
2010 DATA "AFAGA","FALHA","ADAG  
A","HA","AGA","Haja","HALL","GA  
LHAS","FALA","FALHA","AJA","JAL  
"

```

**T**

```

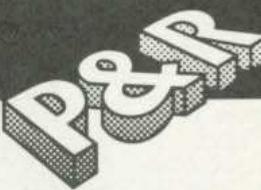
10 OBS$="A S D F G H J K L"
20 CLS
30 DIM WS(28)
40 FOR K=1 TO 28
50 READ WS(K)
60 NEXT
70 PRINT @71,"QUAL O NIVEL DE"
80 PRINT @101,"DIFICULDADE (1-5
) ?"

```

```

90 PRINT @165,"DIGITE (0) PARA  
SAIR"
100 AS=INKEYS:IF AS<"0" OR AS>"  
5" THEN 100
110 A=VAL(AS):ON A+1 GOSUB 999,
200,300,400,600,800
120 ER=0
130 GOTO 70
200 GOSUB 1000
210 AP=1252
220 FOR K=1 TO 9
230 AP=AP+2
240 GOSUB 1100
250 NEXT
260 CLS:PRINT @448,"TEMPO=";TIM  
ER/50;"SEGUNDOS"
270 PRINT"NUMERO DE ERROS=";ER;
280 RETURN
300 GOSUB 1000
310 FOR K=1 TO 20
320 AP=1252+2*RND(9)
330 GOSUB 1100
340 NEXT
350 CLS
360 PRINT @448,"TEMPO=";TIMER/5  
0;"SEGUNDOS"
370 PRINT"NUMERO DE ERROS=";ER;
380 RETURN
400 CLS:PRINT" DIGITE A TECLA  
INDICADA NA TELA"
410 PRINT @206,"      ":";PRINT @23  
8,"      ":";PRINT @270,"      ";
420 FOR K=1 TO 20
430 PS=MIDS(OBS,2*RND(9)-1,1)
440 PRINT @239,PS;
450 AS=INKEYS:IF AS="" THEN 450
460 IF K=1 THEN TIMER=0
470 IF AS<>PS THEN SOUND 5,1:ER  
=ER+1:GOTO 450
480 POKE 1263,128
490 SOUND 200,1
500 NEXT
510 GOTO 350
600 CLS:PRINT" DIGITE A TECLA Q  
UE APARECER"
610 T=0
620 FOR K=1 TO 10
630 PS=WS(RND(28))
640 T=T+LEN(PS)
650 PRINT @237,PS
660 P=1
670 POKE 1228+P,106
680 AS=INKEYS:IF AS="" THEN 680
690 IF K=1 THEN TIMER=0
700 IF AS<>MIDS(PS,P,1) THEN ER  
=ER+1:SOUND 5,1:GOTO 670
710 POKE 1228+P,96
720 SOUND 200,1:P=P+1:IF P<=LEN  
(PS) THEN 670
730 NEXT
740 CLS
750 PRINT @416,USING"PALAVRAS P  
OR MINUTO=###.##";T*500/TIMER
760 GOTO 360
800 CLS:PS="":FOR K=1 TO 6
810 PS=PS+WS(RND(28))+"
820 NEXT
830 PS=LEFTS(PS,LEN(PS)-1)
840 PRINT" DIGITE ESTAS PALAVRA  
S"
850 P=1:PRINT @224,PS
860 AS=INKEYS:IF AS="" THEN 860
870 TIMER=0:GOTO 890

```



Qual a velocidade de datilografia que  
devo tomar como meta para começar?

Na etapa inicial do curso, o mais correto é preocupar-se com a precisão e não com a velocidade. Depois que se consegue datilografar todas as palavras sem cometer nenhum erro, pode-se pensar em acelerar o ritmo de trabalho.

Deve-se praticar os Níveis 1, 2 e 3 até que se consiga datilografar vinte letras em doze ou treze segundos; nos Níveis 4 e 5, deve-se ter como meta cerca de quinze palavras ou mais por minuto.

```

880 AS=INKEYS:IF AS="" THEN 880
890 IF AS<>MIDS(PS,P,1) THEN ER  
=ER+1:SOUND 5,1:GOTO 880
900 POKE 1311+P,PEEK(1247+P)
910 SOUND 200,1:P=P+1:IF P<=LEN  
(PS) THEN 880
920 T=LEN(PS):GOTO 740
999 CLS:END
1000 CLS:PRINT"PRESSIONE A TECL  
A INDICADA PELO ASTERISCO"
1010 PRINT @262,OBS
1020 FOR K=1 TO 1000:NEXT
1030 RETURN
1100 POKE AP,106
1110 AS=INKEYS:IF AS="" THEN 11  
10
1120 IF K=1 THEN TIMER=0
1125 PRINT@14*32,ASC(AS);
1126 PRINT@15*32,PEEK(AP+32);
1130 IF (ASC(AS))<>PEEK(AP+32)T  
HEN SOUND 5,1:ER=ER+1:GOTO 1110
1140 SOUND 200,1
1150 POKE AP,.96
1160 RETURN
9000 DATA FADA,GALA,SALA,DADA,A  
SA,LAKA,ALAGA,ALADA,LA,GAS,DALL  
AS,SAGA,GALHADA,FALA
9010 DATA AFAGA,FALHA,ADAGA,HA,  
AGA,HAJA,JA,HALL,KA,FA,AJA,DAS,  
GALHAS,JAL

```



```

5 R=RND(-TIME)
10 OBS$="A S D F G H J K L C":Z$  
=CHR$(219):SS="L10 O2 G"
20 SCREEN1:CLS
30 DIM WS(28)
40 FOR K=1 TO 28
50 READ WS(K)
60 NEXT
70 LOCATE 7,4:PRINT"Qual nível  
de"
80 LOCATE 5,6:PRINT"dificuldade"

```

```

? (1-5)
90 LOCATE 3,9:PRINT"Tecla <0> para terminar."
100 AS=INKEY$:IF AS<"0" OR AS>"5" THEN 100
110 ON VAL(AS)+1 GOSUB 999,200,
300,400,600,800
120 ER=0
130 GOTO 70
200 GOSUB 1000
210 AP=357
220 FOR K=1 TO 10
230 AP=AP+2
240 GOSUB 1100
250 NEXT
260 CLS
270 LOCATE 5,12:PRINT"Tempo =";INT(TIME/60);"segundos."
280 LOCATE 6:PRINT"Número de erros =";ER;
290 RETURN
300 GOSUB 1000
310 FOR K=1 TO 20
320 AP=359+INT(RND(1)*10)*2
330 GOSUB 1100
340 NEXT
350 GOTO 260
400 CLS:PRINT"Digite a letra mostrada na tela:"
410 LOCATE 11,9:PRINTZ$;Z$;Z$;Z$:LOCATE 11:PRINTZ$;" ";Z$:LOCATE11:PRINTZ$;" ";Z$:LOCATE11:PRINTZ$;Z$;Z$;Z$;
420 FOR K=1 TO 20
430 P$=MIDS(OBS,INT(RND(1)*10)*2+1,1)
440 LOCATE 13,11:PRINTPS;
450 AS=INKEY$:IF AS=""THEN450
460 IF K=1 THEN TIME=0
470 IF AS<>P$ THEN PLAY SS:ER=E
R+1:GOTO450
480 PRINTCHR$(8);CHR$(32);
490 BEEP
500 NEXT:GOTO 260
600 CLS:PRINT"Digite a palavra que aparecer:"
610 TL=0
620 FOR K=1 TO 10
630 P$=W$(INT(RND(1)*28)+1)
640 T=LEN(P$):TL=TL+T:L=INT((30-T)/2)
645 LOCATE 0,11:PRINTSPACES(30)
;
650 LOCATE L,11:PRINTPS
655 FOR J=1 TO T
660 LOCATE L,10:PRINTCHR$(205);
670 AS=INKEY$:IF AS=""THEN670
680 IF K=1 THEN TIME=0
690 IF AS<>MIDS(P$,J,1)THEN PLAY SS:ER=ER+1:GOTO 670
700 LOCATE L,10:PRINTCHR$(32);
710 BEEP:L=L+1:NEXT
720 NEXT
730 CLS:LOCATE 2,16:PRINT "Palavras por minuto:";:PRINT USING
"###.##";TL*600/TIME
740 GOTO 270
800 CLS:P$=""
810 FOR K=1 TO 5:P$=P$+WS(INT(RND(1)*28)+1)+CHR$(32):NEXT
820 P$=LEFT$(P$,LEN(P$)-1)
830 PRINT"Digite estas palavras :"
840 P=1:LOCATE 0,11:PRINTPS
850 AS=INKEY$:IF AS=""THEN850
860 TIME=0:GOTO 880
870 AS=INKEY$:IF AS=""THEN870
880 IF AS<>MIDS(P$,P,1)THEN PLAY SS:ER=ER+1:GOTO 870
890 LOCATE-1+P,14:PRINTMIDS(P$,P,1)
900 BEEP:P=P+1:IF P<=LEN(P$) THEN 870
910 TL=LEN(P$):GOTO 730
999 SCREEN0:END
1000 CLS:PRINT"Digite o caractere indicado pela seta:"
1010 LOCATE 5,12:PRINTOBS
1020 FOR K=1 TO 1000:NEXT
1030 RETURN
1100 VPOKE BASE(5)+AP,205
1110 AS=INKEY$:IF AS=""THEN 1110
1120 IF K=1 THEN TIME=0
1130 IF ASC(AS)<>VPEEK(BASE(5)+32+AP)THEN PLAY SS:ER=ER+1:GOTO 1110
1140 BEEP
1150 VBOKE BASE(5)+AP,0

```



```

1160 RETURN
9000 DATA FADA,SALA,ADAGA,DADA,
FAÇA,FA,GALA,HAJA,AGA,HALL,ALAD
A,ASA,GAS,FALHA
9010 DATA FALA,LACA,ALAGA,LAKA,
SAGA,JA,AJA,AFAGA,GALHADA,DALLA
S,LA,HA,ASSA,LAJA

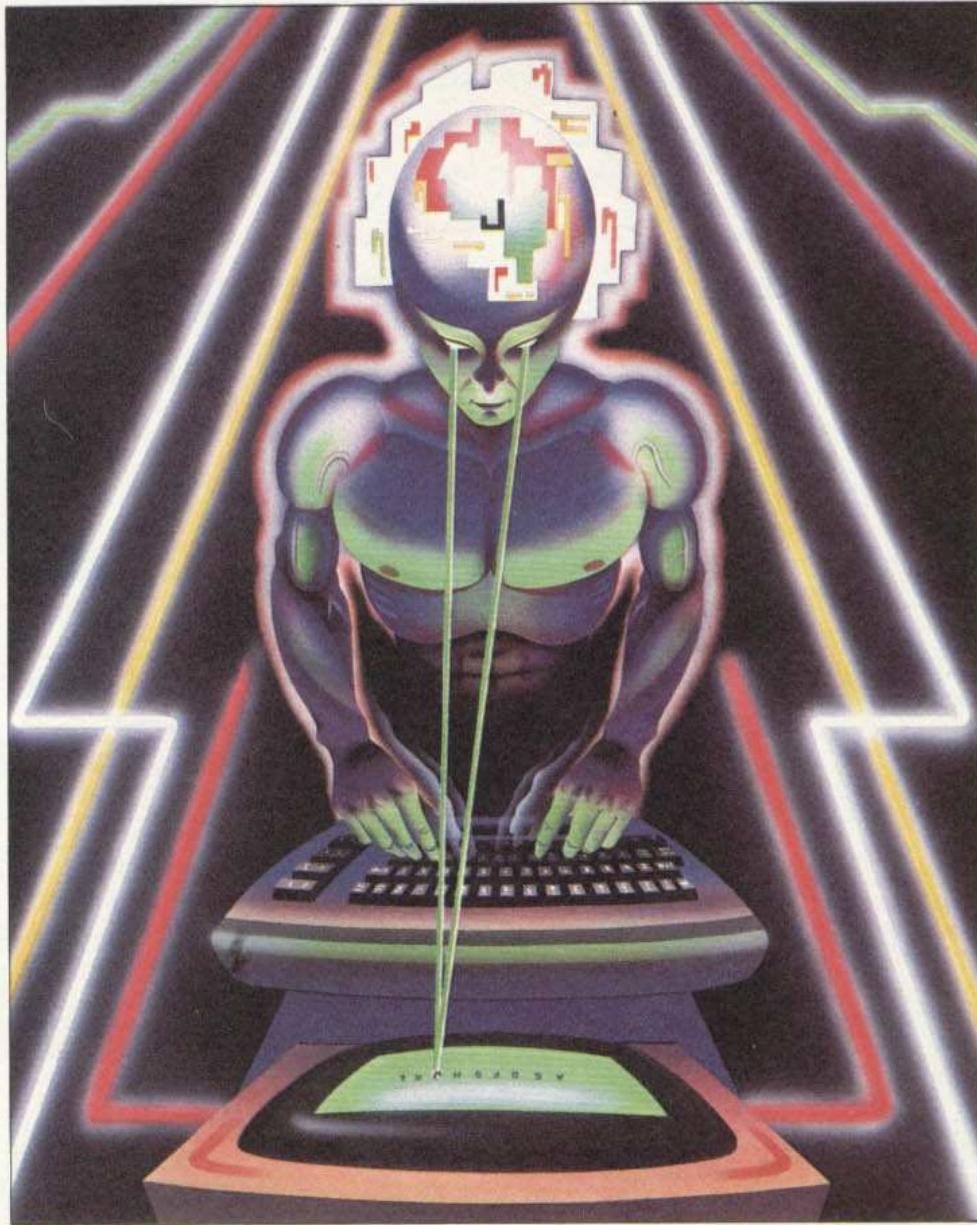
```



```

10 OB$ = "A S D F G H J
K L ;"
20 HOME
30 DIM WS(28)
40 FOR K = 1 TO 28: READ WS(K)
: NEXT
50 HTAB 14: VTAB 5: PRINT "Qua
1 nivel de"
60 PRINT : HTAB 12: PRINT "dif
iculdade? (1-5)"
70 HTAB 9: VTAB 11: PRINT "Tec
le <0> para terminar."
80 GET AS: IF AS < "0" OR AS >
"5" THEN 80
90 ON VAL (AS) + 1 GOSUB 999.
200,300,400,500,600
100 ER = 0
110 GOTO 50
200 GOSUB 1000
210 AP = 2
220 FOR K = 1 TO 10:AP = AP +
3: GOSUB 1100: NEXT
230 HOME
240 VTAB 15: HTAB 12: PRINT "N
umero de erros = ";ER;
250 RETURN
300 GOSUB 1000
310 FOR K = 1 TO 20
320 AP = 2 + INT ( RND (1) * 1
0 + 1) * 3
330 GOSUB 1100: NEXT
340 GOTO 230
400 HOME : PRINT "Digite a let
ra mostrada na tela:"
410 VTAB 9: HTAB 18: INVERSE :
PRINT " ":" HTAB 18: PRINT "
": HTAB 18: PRINT "
420 FOR K = 1 TO 20:P$ = MIDS
(OB$, INT ( RND (1) * 10 + 1)
* 3 - 2,1)
430 VTAB 10: HTAB 19: PRINT P$
; CHR$ (8);
440 GET AS: IF AS < > PS THEN
PRINT CHR$ (7);:ER = ER + 1:
GOTO 440
450 A = PEEK (- 16336):A = P
EEK (- 16336): NEXT : NORMAL :
GOTO 230
500 HOME : PRINT "Digite a pal
avra que aparecer:"
510 TL = 0
520 FOR K = 1 TO 10
530 PS = WS( INT ( RND (1) * 28
) + 1)
540 T = LEN (PS):L = INT ((40
- T) / 2)
550 PRINT : VTAB 11: CALL - 9
58: HTAB L: PRINT PS
560 FOR J = 0 TO T - 1: VTAB 1
0: HTAB J + L: PRINT "*"; CHR$(
8);
570 GET AS: IF AS < > MIDS (
PS,J + 1,1) THEN PRINT CHR$ (

```



```

7);:ER = ER + 1: GOTO 570
580 A = PEEK (- 16336):A = P
EEK (- 16336): PRINT CHR$ (32
);: NEXT
585 NEXT
590 GOTO 230
600 HOME :PS = ""
610 FOR K = 1 TO 6:PS = PS + W
$( INT ( RND (1) * 28) + 1) +
CHR$ (32): NEXT
620 PS = LEFT$ (PS, LEN (PS) -
1)
630 PRINT "Digite estas palavr
as:"
640 P = 1: VTAB 11: HTAB 4: PRI
NT PS
650 GET AS: IF AS < > MIDS (
PS,P,1) THEN PRINT CHR$ (7);:
ER = ER + 1: GOTO 650
660 VTAB 15: HTAB P + 3: PRINT
MIDS (PS,P,1)
670 A = PEEK (- 16336):A = P
EEK (- 16336):P = P + 1: IF P
< = LEN (PS) THEN 650
680 GOTO 230
999 HOME : END
1000 HOME : PRINT "Digite o ca
racter indicado pelo a
sterisco:"
1010 VTAB 12: HTAB 5: PRINT OB
$ 
1020 FOR K = 1 TO 500: NEXT
1030 RETURN
1100 POKE 1319 + AP,170
1110 GET AS: IF ASC (AS) < >
PEEK (1319 + AP + 128) - 128
THEN PRINT CHR$ (7);:ER = ER
+ 1: GOTO 1110
1120 A = PEEK (- 16336):A = P
EEK (- 16336): POKE 1319 + AP
,160
1130 RETURN
9000 DATA FALA,AGA,AFAGA,GAS,
SALA,DADA,HAJA,HALL,LAKA,ASA,AL
ADA,ASDFG,SAGA,KAKA
9010 DATA AJA,FA,ALAGA,JA,AGA
,HA,GALHADA,FALHA,GALA,FADA,DAL
LAS,ASSADA,LALA,;LKJH

```

# CÓDIGO DE CONTROLE

Os códigos de controle correspondem, na maioria dos micros, aos códigos de 0 a 31 no sistema ASCII, e servem para realizar uma série de funções relacionadas principalmente com o controle do vídeo.

Caso você não saiba ainda, ASCII é um sistema de codificação padronizado mundialmente e utilizado por quase todos os computadores. ASCII significa *American Standard Code for Information Interchange* (Código Padrão Americano para Intercâmbio de Informação), e se pronuncia *ásqui*.

Originalmente, os códigos de controle do ASCII correspondiam às funções básicas de transmissão de um terminal ligado ao computador: início de transmissão, sinal de atenção, fim de transmissão, acionamento da campainha do terminal, etc.

Entretanto, cada fabricante de microcomputadores aproveitou a faixa de códigos de 0 a 31 para implementar funções diferentes. Com poucas exceções, portanto, a maioria dos códigos de controle não é padronizada, como acontece com os códigos correspondentes a caracteres comuns (letras, números, sinais matemáticos e de pontuação, etc.).

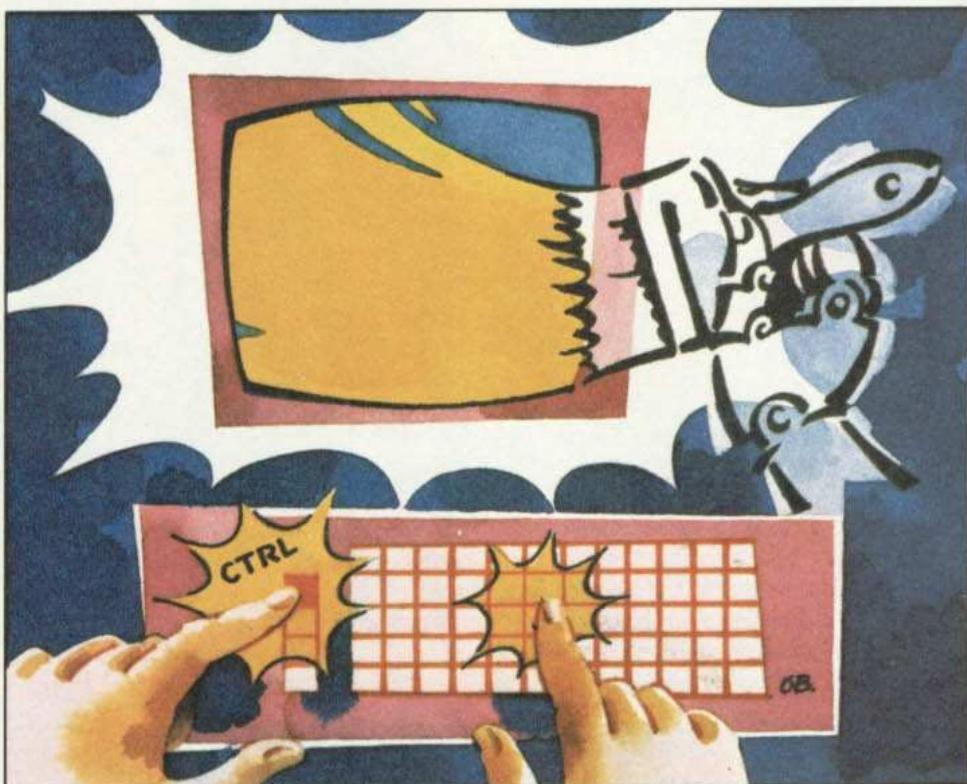
## O ACESSO AOS CÓDIGOS DE CONTROLE

Existem duas técnicas básicas de utilização de códigos de controle. Uma delas — mais universal — pode ser aplicada a qualquer microcomputador que disponha da função **CHR\$** no interpretador BASIC. A outra é aplicável apenas a alguns computadores: é a entrada direta dos códigos de controle por intermédio do teclado. A tecla rotulada **CONTROL** (ou ainda **CTRL**), existente em muitos micros, deve ser pressionada em combinação com outras teclas, para introduzir o código de controle numa linha de comando ou de programa.

A função **CHR\$** (abreviatura de *character*) permite converter um código numérico inteiro, situado entre 0 e 255, ao seu caractere correspondente. Por exemplo, **PRINT CHR\$(65)** escreverá na tela a letra A (maiúscula).

A função **CHR\$** funciona também com os caracteres de controle. Dessa

Códigos de controle podem ser usados no lugar de muitos comandos em BASIC. Entrados diretamente por meio do teclado em certos micros, eles proporcionam uma ação imediata.



forma, se acionarmos um comando **PRINT** associado a um ou mais caracteres de controle, poderemos realizar uma série de tarefas básicas (até mesmo algumas que normalmente podem não ser acessíveis por intermédio de comandos BASIC). A seguir, damos alguns exemplos para o TRS-80 (os outros computadores serão abordados na continuação deste capítulo).

## T

Você já conhece a instrução **CLS**, que serve para limpar a tela e retornar o cursor à posição superior esquerda do vídeo. Não seria interessante se pudéssemos apagar a tela apenas de um determinado ponto para baixo ou até o final de uma linha? Efeitos desse tipo são facilmente obtidos por meio da combinação do comando **PRINT CHR\$(n)** com os seguintes códigos:

- 8 - recua e apaga o último caractere na tela;
- 10 - avança o cursor para o começo

- da linha seguinte;
- 14 - liga o cursor;
- 15 - desliga o cursor;
- 23 - converte a tela para a largura de 32 caracteres;
- 28 - retorna o cursor à posição 0,0, sem apagar a tela;
- 29 - move o cursor para o começo da linha;
- 30 - apaga da posição do cursor até o final da linha;
- 31 - apaga o espaço que vai da posição do cursor até o final da tela.

Uma sequência de caracteres de controle pode ser colocada também dentro de uma cadeia de caracteres, por meio da função **STRING\$**. Por exemplo: **PRINT STRING\$(5,10)** provoca o avanço do cursor cinco linhas para baixo. Esse comando corresponde à ação de pressionar a tecla com a flecha para baixo cinco vezes seguidas, com a vantagem de poder ser colocado dentro de um programa (corresponde a cinco comandos **PRINT** em branco).

| LINHA             | FABRICANTE       | MODELO          |
|-------------------|------------------|-----------------|
| Apple II+         | Appletronica     | Thor 2010       |
| Apple II+         | CCE              | MC-4000 Exato   |
| Apple II+         | CPA              | Absolutus       |
| Apple II+         | CPA              | Polaris         |
| Apple II+         | Digitus          | DGT-AP          |
| Apple II+         | Dismac           | D-8100          |
| Apple II+         | ENIAC            | ENIAC II        |
| Apple II+         | Franklin         | Franklin        |
| Apple II+         | Houston          | Houston AP      |
| Apple II+         | Magnex           | DM II           |
| Apple II+         | Maxitronica      | MX-2001         |
| Apple II+         | Maxitronica      | MX-48           |
| Apple II+         | Maxitronica      | MX-64           |
| Apple II+         | Maxitronica      | Maxitronic I    |
| Apple II+         | Microcraft       | Craft II Plus   |
| Apple II+         | Milmar           | Apple II Plus   |
| Apple II+         | Milmar           | Apple Master    |
| Apple II+         | Milmar           | Apple Senior    |
| Apple II+         | Omega            | MC-400          |
| Apple II+         | Polymax          | Maxxi           |
| Apple II+         | Polymax          | Poly Plus       |
| Apple II+         | Spectrum         | Microengenho I  |
| Apple II+         | Spectrum         | Spectrum ed     |
| Apple II+         | Suporte          | Venus II        |
| Apple II+         | Sycomig          | SIC I           |
| Apple II+         | Unitron          | AP II           |
| Apple II+         | Victor do Brasil | Elppa II Plus   |
| Apple II+         | Victor do Brasil | Elppa Jr.       |
| Apple IIe         | Microcraft       | Craft IIe       |
| Apple IIe         | Microdigital     | TK-3000 IIe     |
| Apple IIe         | Spectrum         | Microengenho II |
| MSX               | Gradiente        | Expert GPC-1    |
| MSX               | Sharp            | Hotbit HB-8000  |
| Sinclair Spectrum | Microdigital     | TK-90X          |
| Sinclair Spectrum | Timex            | Timex 2000      |
| Sinclair ZX-81    | Apply            | Apply 300       |
| Sinclair ZX-81    | Engebras         | AS-1000         |
| Sinclair ZX-81    | Filcres          | NEZ-8000        |
| Sinclair ZX-81    | Microdigital     | TK-82C          |
| Sinclair ZX-81    | Microdigital     | TK-83           |
| Sinclair ZX-81    | Microdigital     | TK-85           |
| Sinclair ZX-81    | Prologica        | CP-200          |
| Sinclair ZX-81    | Ritas            | Ringo R-470     |
| Sinclair ZX-81    | Timex            | Timex 1000      |
| Sinclair ZX-81    | Timex            | Timex 1500      |
| TRS-80 Mod. I     | Dismac           | D-8000          |
| TRS-80 Mod. I     | Dismac           | D-8001/2        |
| TRS-80 Mod. I     | LNW              | LNW-80          |
| TRS-80 Mod. I     | Video Genie      | Video Genie I   |
| TRS-80 Mod. III   | Digitus          | DGT-100         |
| TRS-80 Mod. III   | Digitus          | DGT-1000        |
| TRS-80 Mod. III   | Kemitron         | Naja 800        |
| TRS-80 Mod. III   | Prologica        | CP-300          |
| TRS-80 Mod. III   | Prologica        | CP-500          |
| TRS-80 Mod. III   | Sysdata          | Sysdata III     |
| TRS-80 Mod. III   | Sysdata          | Sysdata Jr.     |
| TRS-80 Mod. IV    | Multix           | MX-Compacto     |
| TRS-80 Mod. IV    | Sysdata          | Sysdata IV      |
| TRS-Color         | Codimex          | CS-6508         |
| TRS-Color         | Dynacom          | MX-1600         |
| TRS-Color         | LZ               | Color 64        |
| TRS-Color         | Microdigital     | TKS-800         |
| TRS-Color         | Prologica        | CP-400          |

## CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR NOME

| FABRICANTE       | MODELO          | PAÍS   | LINHA             |
|------------------|-----------------|--------|-------------------|
| Appletronica     | Thor 2010       | Brasil | Apple II +        |
| Apply            | Apply 300       | Brasil | Sinclair ZX-81    |
| CCE              | MC-4000 Exato   | Brasil | Apple II +        |
| CPA              | Absolutus       | Brasil | Apple II +        |
| CPA              | Polaris         | Brasil | Apple II +        |
| Codimex          | CS-6508         | Brasil | TRS-Color         |
| Digitus          | DGT-100         | Brasil | TRS-80 Mod. III   |
| Digitus          | DGT-1000        | Brasil | TRS-80 Mod. III   |
| Dismac           | D-8000          | Brasil | TRS-80 Mod. I     |
| Dismac           | D-8001/2        | Brasil | TRS-80 Mod. I     |
| Dismac           | D-8100          | Brasil | Apple II +        |
| Dynacom          | MX-1600         | Brasil | TRS-Color         |
| ENIAC            | ENIAC II        | Brasil | Apple II +        |
| Engebras         | AS-1000         | Brasil | Sinclair ZX-81    |
| Filcres          | NEZ-8000        | Brasil | Sinclair ZX-81    |
| Franklin         | Franklin        | USA    | Apple II +        |
| Gradiente        | Expert GPC1     | Brasil | MSX               |
| Houston          | Houston AP      | Brasil | Apple II +        |
| Kemitron         | Naja 800        | Brasil | TRS-80 Mod. III   |
| LNW              | LNW-80          | USA    | TRS-80 Mod. I     |
| LZ               | Color 64        | Brasil | TRS-Color         |
| Magnex           | DM II           | Brasil | Apple II +        |
| Maxitronica      | MX-2001         | Brasil | Apple II +        |
| Maxitronica      | MX-48           | Brasil | Apple II +        |
| Maxitronica      | MX-64           | Brasil | Apple II +        |
| Maxitronic I     | Maxitronic I    | Brasil | Apple II +        |
| Microcraft       | Craft II Plus   | Brasil | Apple II +        |
| Milmar           | Apple II Plus   | Brasil | Apple II +        |
| Milmar           | Apple Master    | Brasil | Apple II +        |
| Milmar           | Apple Senior    | Brasil | Apple II +        |
| Multix           | MX-Compacto     | Brasil | TRS-80 Mod. IV    |
| Omega            | MC-400          | Brasil | Apple II +        |
| Polymax          | Maxxi           | Brasil | Apple II +        |
| Polymax          | Poly Plus       | Brasil | Apple II +        |
| Prologica        | CP-200          | Brasil | Sinclair ZX-81    |
| Prologica        | CP-300          | Brasil | TRS-80 Mod. III   |
| Prologica        | CP-400          | Brasil | TRS-Color         |
| Prologica        | CP-500          | Brasil | TRS-80 Mod. III   |
| Ritas            | Ringo R-470     | Brasil | Sinclair ZX-81    |
| Sharp            | Hotbit HB-8000  | Brasil | MSX               |
| Spectrum         | Microengenho I  | Brasil | Apple II +        |
| Spectrum         | Microengenho II | Brasil | Apple IIe         |
| Spectrum         | Spectrum ed     | Brasil | Apple II +        |
| Suporte          | Venus II        | Brasil | Apple II +        |
| Sycomig          | SIC I           | Brasil | Apple II +        |
| Sysdata          | Sysdata III     | Brasil | TRS-80 Mod. III   |
| Sysdata          | Sysdata IV      | Brasil | TRS-80 Mod. IV    |
| Sysdata          | Sysdata Jr.     | Brasil | TRS-80 Mod. III   |
| Timex            | Timex 1000      | USA    | Sinclair ZX-81    |
| Timex            | Timex 1500      | USA    | Sinclair ZX-81    |
| Timex            | Timex 2000      | USA    | Sinclair Spectrum |
| Unitron          | AP II           | Brasil | Apple II +        |
| Victor do Brasil | Elppa II Plus   | Brasil | Apple II +        |
| Victor do Brasil | Elppa Jr.       | Brasil | Apple II +        |
| Video Genie      | Video Genie I   | USA    | TRS-80 Mod. I     |

UM LOGOTIPO PARA CADA MODELO DE COMPUTADOR



Sinclair ZX-81



TRS-80



TK-2000



Spectrum



TRS-Color



Apple II



MSX

**INPUT** foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.

Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

# NO PRÓXIMO NÚMERO

## APLICAÇÕES

Continue seu curso de datilografia com o computador, praticando agora com todo o alfabeto.

## PROGRAMAÇÃO DE JOGOS

Aprenda a movimentar um herói em meio às ameaças de um mundo hostil cujos limites são os da imaginação.

## PROGRAMAÇÃO BASIC

Como examinar a memória do computador. Os comandos **PEEK** e **POKE**. Cores do TRS-Color.

CURSO PRÁTICO 14 DE PROGRAMAÇÃO DE COMPUTADORES

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

Cz\$ 20,00



PEEK 12460

